

■ PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS
AND
VOLTS

NUTS AND VOLTS

www.nutsvolts.com
April 2014

EVERYTHING FOR ELECTRONICS

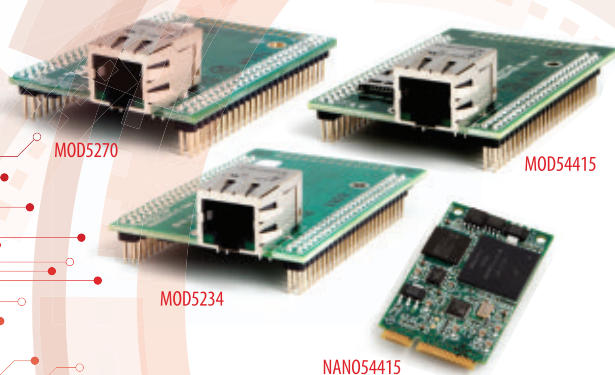
H Full Motion
HOME Flight
SIMULATOR

Go "Virtually"
Anywhere!



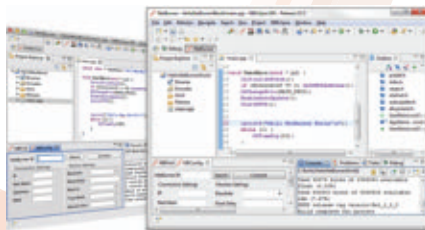
◆ The BaTESLA Coil
◆ Decade Box Revisited

Ethernet Core Modules with High-Performance Connectivity Options



- **MOD5270**
147.5 MHz processor with 512KB Flash & 8MB RAM · 47 GPIO · 3 UARTs · I²C · SPI
- **MOD5234**
147.5 MHz processor with 2MB flash & 8MB RAM · 49 GPIO · 3 UARTs · I²C · SPI · CAN · eTPU (for I/O handling, serial communications, motor/timing/engine control applications)
- **MOD54415**
250 MHz processor with 32MB flash & 64MB RAM · 42 GPIO · 8 UARTs · 5 I²C · 3 SPI · 2 CAN · SSI · 8 ADC · 2 DAC · 8 PWM · 1-Wire® interface
- **NANO54415**
250 MHz processor with 8MB flash & 64MB RAM · 30 GPIO · 8 UARTs · 4 I²C · 3 SPI · 2 CAN · SSI · 6 ADC · 2 DAC · 8 PWM · 1-Wire® interface

Add Ethernet connectivity to an existing product, or use it as your product's core processor



The goal: Control, configure, or monitor a device using Ethernet

The method: Create and deploy applications from your Mac or Windows PC. Get hands-on familiarity with the NetBurner platform by studying, building, and modifying source code examples.

The result: Access device from the Internet or a local area network (LAN)

The NetBurner Ethernet Core Module is a device containing everything needed for design engineers to add network control and to monitor a company's communications assets. For a very low price point, this module solves the problem of network-enabling devices with 10/100 Ethernet, including those requiring digital, analog and serial control.

MOD5270-100IR.....\$69 (qty. 100)	NNDK-MOD5270LC-KIT\$99
MOD5234-100IR.....\$99 (qty. 100)	NNDK-MOD5234LC-KIT\$249
MOD54415-100IR.....\$89 (qty. 100)	NNDK-MOD54415LC-KIT\$129
NANO54415-200IR...\$69 (qty. 100)	NNDK-NANO54415-KIT.....\$99

NetBurner Development Kits are available to customize any aspect of operation including web pages, data filtering, or custom network applications. The kits include all the hardware and software you need to build your embedded application.

➤ **For additional information please visit**
<http://www.netburner.com/kits>



PIC[®]clicker

price:
\$19⁰⁰



This amazingly compact starter development kit brings dozens of Click[™] add-on boards that will inspire you to make great projects.

New idea is just a click away!

20 Build the BaTESLA Coil

How would of Nikola Tesla designed his famous coil if he had been able to access modern day electronic parts? Here's one idea.

■ By Matt Bates

28 The Dryer Minder

While originally created for a specific purpose, this basic "notification" circuit can be implemented in many different applications.

■ By Jim Lacenski

34 The Decade Box Revisited

Decade boxes allow users to dial in precise amounts of electrical resistance to be inserted into a circuit they're designing. Original boxes had faceplates with the values printed on them, but extra math was always involved. Here's a handy update to that reliable piece of equipment.

■ By Frank Muratore

38 A Full Motion Home Simulator

Go "virtually" anywhere with your own homebrew flight simulator.

■ By Walt Noon

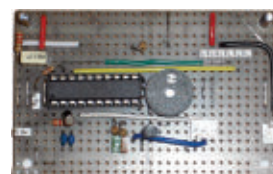
46 MakerPlot — The DIY Software Kit ... Part 7

Be introduced to MakerPlot's best kept secret: the Logs(Debug) Immediate window.

■ By John Gavlik and Martin Hebel



Page 34



Page 28

Page 20

Departments

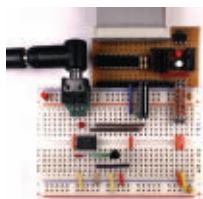
06	DEVELOPING PERSPECTIVES	19	SHOWCASE
	<i>A Maker's Dozen: Designing Around Failure</i>	67	ELECTRO-NET
07	READER FEEDBACK	74	NV WEBSTORE
16	NEW PRODUCTS	77	CLASSIFIEDS
		78	TECH FORUM
		81	AD INDEX

08 PICAXE Primer

Sharpening Your Tools of Creativity

PICAXE-Pi Communications — Part 2.

We'll continue with our PICAXE-Pi communications experiments, and cover the basic techniques that can be used to send a PICAXE ADC reading to the Pi from any analog sensor we choose.



56 Smiley's Workshop

Programming • Hardware • Projects

The Arduino Classroom.

Arduino 101/Chapter 4: Digital Input ...

Pushbuttons.

Learn how to design circuits with pushbuttons and how to utilize them to get your system to take action when they are pressed (or not).

68 The Design Cycle

Advanced Techniques for Design Engineers

Total Eclipse of the Cross Compiler.

The BeagleBone Black and Raspberry Pi are different in many ways. They are also alike in many ways. However, when it comes to programming them, the equalizer appears in the form of a Linux-based GNU cross compiler toolchain running under the umbrella of an Eclipse IDE.

52 Open Communication

The Latest in Networking and Wireless Technologies

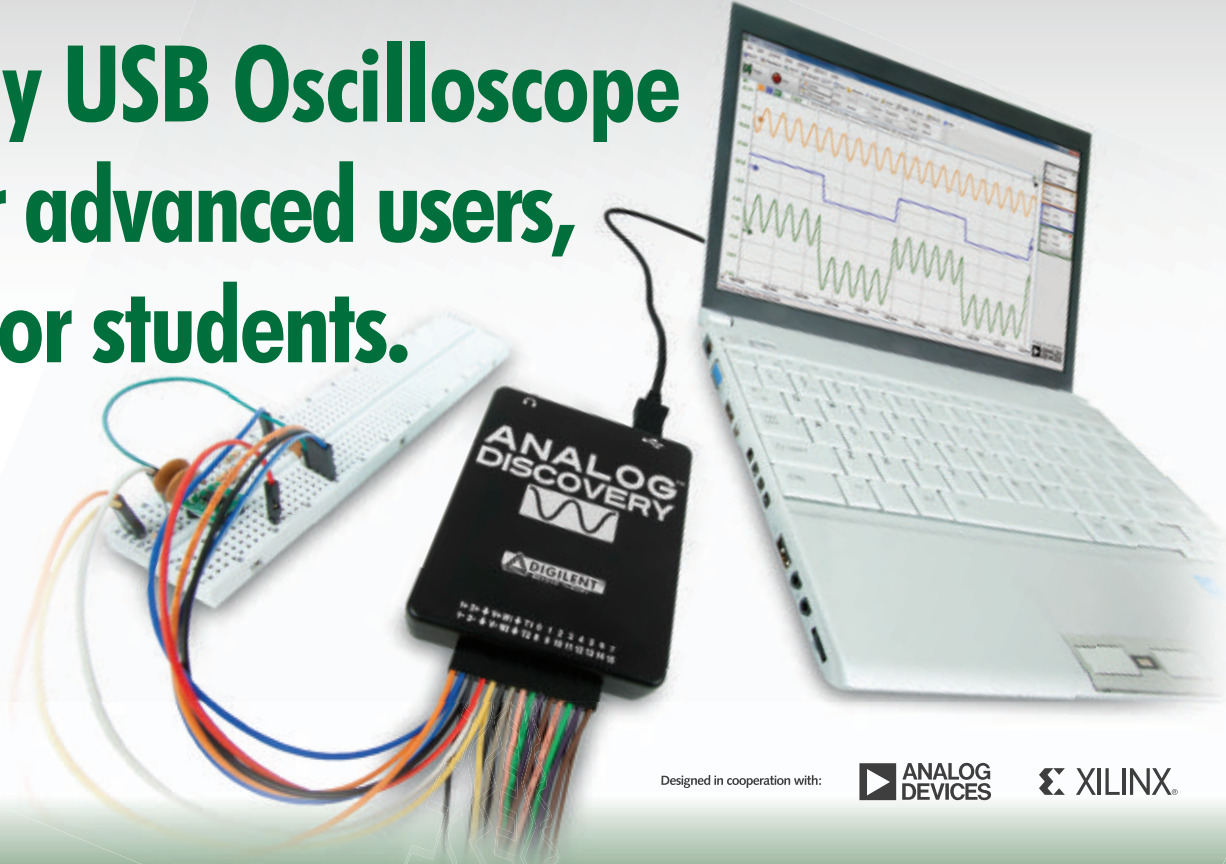
Software-Defined Radios Everywhere.

Virtually all radios today — cell phones and Wi-Fi WLAN, for example — are software-defined radios. Find out exactly what SDRs are and how they work.

Columns

The only USB Oscilloscope built for advanced users, priced for students.

\$99.00
(U.S. student price)



Designed in cooperation with:



Turn any PC into a powerful electrical engineering workstation! The USB-powered Analog Discovery lets you measure, visualize, analyze, record and control mixed signal circuits of all kinds. It's small enough to fit in your pocket, but powerful enough to replace a stack of lab equipment. Driven by the free WaveForms™ software, the Analog Discovery lets you build and test analog and digital circuits in virtually any environment, in or out of the lab.

- 2-Channel Oscilloscope
- 2-Channel Waveform Generator
- 16-Channel Logic Analyzer
- 16-Channel Digital Pattern Generator
- $\pm 5\text{VDC}$ Power Supplies
- Spectrum Analyzer
- Network Analyzer
- Voltmeter
- Digital I/O

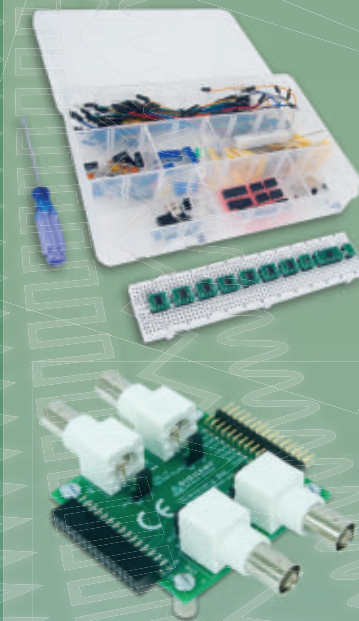
Now supported by MATLAB / MATLAB student edition

Also Available:

Analog Parts Kit (APK)

- 20+ ICs from Analog Devices
- 200+ discrete components
- Solderless breadboard
- Perfect for Circuits 1&2 classes

\$69.99



Discovery BNC Adapter

- Allows the use of standard BNC-terminated probes
- Selectable AC & DC coupling

\$19.99



www.digilentinc.com/AnalogDiscovery





by Bryan Bergeron, Editor

DEVELOPING PERSPECTIVES

A MAKER'S DOZEN: Designing Around Failure

I just finished a major production run of simulators for training physicians on how to examine the eye. I needed to deliver 10 units, so I started with 13 mannequins — each with high-resolution LCD screens for the backs of the eyes (retinae), an Arduino, and a custom interface board. In addition, I printed holders and mounting brackets for the LCD and each of the two boards. Add to that about 30 wire-wrap connections for power, an LED indicator, and interconnecting the major boards, and you have a recipe for guaranteed failure. The only question was the success rate.

For this project, there were several sources of failure. For starters, two of the custom boards were DOA. It turned out the manufacturer used an NPN transistor in the output circuit when a PNP was called for. I had to contact the manufacturer and arrange for replacements. Not a major problem, but it took several days for the supplier to provide the replacement parts. Then, there was operator error. I flipped a green and blue pair when wire-wrapping the LCD to the Arduino on one of the simulators. With a wire-wrap tool, this was an easy quick fix.

Slightly more problematic was my MakerGear M2. It performed superbly when printing the circuit board holders in

PLA, but when I switched to ABS plastic, output quality became erratic. After a bit of experimenting, I found that increasing the platform temperature a few degrees solved the problem. As I've learned on several models of 3D printers, these problems are to be expected. Still, each failure cost time — each component required two to three hours to print. As it was, the printer was cranking nearly 24/7 for a week to make the delivery deadline.

In the end, I delivered 10 units on time. Within a week of that, I had repaired the additional three units, which I maintain for backup and quick replacement. Based on my failure rate, I'd say that a proper maker's dozen is 13.5 units. That is, if you need to make a dozen relatively complex devices, order enough spare parts for 1.5 additional units.

You might be wondering how any business could succeed with such a high failure rate. After all, if one in 10 or so consumer electronics products failed, then there wouldn't be much of a consumer electronics industry. A difference between DIY and commercial electronics is that DIY components are often seconds and the overall design specifications shift over time.

Consider the 3D printer, for example. After the first print of a given component, I inevitably modified the model to improve it. Adding a brace to a thin wall, moving a mounting hole, or simply changing the color of the PLA filament has

risks. In addition, not all PLA is created equal. I've found that the inexpensive bulk PLA on eBay simply isn't as good as the more expensive PLA filament from the print manufacturer. I don't know if it's the chemical composition, the diameter, or variation in some other parameter, but I do know that the PLA from MakerGear produces consistently better prints.

If you're building one-offs, you might not notice this relatively high failure rate. It's worth considering if you're building a number of units for, say, a club or classroom. Plus, when it comes to DIY, it's not really a "failure rate" but a learning opportunity. After all, you're not going to learn much if your experiments always work. On the contrary, if you're DIYing is 100% successful, then you're not pushing the edge hard enough. Good luck with all your DIY projects. **NV**

DS1000Z
Digital Oscilloscope
 Advanced Analysis for Entry Level Applications!

- 70MHz & 100MHz Models
- 1GSa/s Max. Sample Rate
- Serial Triggering & Decode
- Integrated 25MHz Generator
- Models Starting at \$585

Visit RigolScope.com
or call 877-4-RIGOL-1

Compare vs. TEK DPO2000 & Save!

Published Monthly By
T & L Publications, Inc.

430 Princeland Ct.
Corona, CA 92879-1300

(951) 371-8497

FAX (951) 371-3052

Webstore orders only **1-800-783-4624**

www.nutsvolts.com

Subscriptions

Toll Free **1-877-525-2539**

Outside US **1-818-487-4545**

P.O. Box 15277

North Hollywood, CA 91615

FOUNDER

Jack Lemieux

PUBLISHER

Larry Lemieux

publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES

Robin Lemieux

robin@nutsvolts.com

EDITOR

Bryan Bergeron

techedit-nutsvolts@yahoo.com

CONTRIBUTING EDITORS

Joe Pardue

Fred Eady

Lou Frenzel

Jim Lacenski

Walt Noon

John Gavlik

Ron Hackett

Matt Bates

Frank Muratore

CIRCULATION DEPARTMENT

subscribe@nutsvolts.com

SHOW COORDINATOR

Audrey Lemieux

MARKETING COORDINATOR WEBSTORE

Brian Kirkpatrick

sales@nutsvolts.com

WEB CONTENT

Michael Kaudze

website@nutsvolts.com

ADMINISTRATIVE ASSISTANT

Debbie Stauffacher

PRODUCTION

Sean Lemieux

Copyright © 2014 by T & L Publications, Inc.
All Rights Reserved

All advertising is subject to publisher's approval. We are not responsible for mistakes, misprints, or typographical errors. *Nuts & Volts Magazine* assumes no responsibility for the availability or condition of advertised items or for the honesty of the advertiser. The publisher makes no claims for the legality of any item advertised in *Nuts & Volts*. This is the sole responsibility of the advertiser. Advertisers and their agencies agree to indemnify and protect the publisher from any and all claims, action, or expense arising from advertising placed in *Nuts & Volts*. Please send all editorial correspondence, UPS, overnight mail, and artwork to: 430 Princeland Court, Corona, CA 92879.

Printed in the USA on SFI & FSC stock.



READER FEEDBACK

Bias on Production Run

Regarding Ryan Clarke's January 2014 article on building a PIC-based remote temperature sensor, I checked the calculation of the base bias resistor for Q1. Clarke apparently used the minimum battery voltage (2.7V), the saturation voltage of the transistor (0.75V), the Beta value of 10 from the sidebar, and the measured minimum I_c value of 3.87 mA for the RF module.

The resulting 5,039 ohms will indeed keep the 2N3904 in saturation for all battery voltages ... for that particular RF module.

However, if you were designing this circuit for production, you'd have to plan for some RF modules to draw the specified minimum I_c current (11 mA). That would yield a bias resistor value of 1,772 ohms, or

1.6K ohms as the closest E24 resistance value. This would admittedly reduce the battery life a little, but that's better than having some circuits fail due to variations in the RF module I_c current.

David Naegle

Absolutely, and my first "go around" with the design was built that way. However, this is not a production design, but one tailored to what was on hand. So, I got more specific on the resistor as you see. Thank you for your feedback. I appreciate it.

Ryan Clarke



EARN MORE MONEY

Get your dream job!

Be an FCC Licensed Wireless Technician!

Make up to \$100,000 a year and more with NO college degree

Get your "FCC Commercial License" with our proven Home-Study Course!

- No costly school. No classes to attend.
- Learn at home. Low cost!
- No experience needed!
- **MONEY BACK GUARANTEE:** You get your FCC License or money refunded!



Move to the front of the employment line in Radio-TV, Communications, Avionics, Radar, Maritime and more... even start your own business!

Call now for FREE info: 800-877-8433 ext. 109

www.LicenseTraining.com

COMMAND PRODUCTIONS FCC License Training
Industrial Center, 480 Gate Five Rd., PO Box 3000, Sausalito, CA 94966-3000

PICAXE-Pi Communications —

Part 2

In this month's Primer, we're going to continue our PICAXE-Pi serial communication experiments. As you probably already know, there aren't any ADC (analog-to-digital converter) inputs available on the Pi, but PICAXE processors have plenty of them: three on the 08M2; seven on the 14M2; 10 on the 18M2; and 11 on the 20M2 and 20X2. To demonstrate one way that we can transfer an ADC reading from a PICAXE processor to the Pi, we're going to interface an MCP9700A (a.k.a., 9700A) analog temperature sensor with an 08M2 processor, and serially send the sensor's ADC reading on to the Pi for display. Of course, we can also do the same thing with any PICAXE processor, and with a variety of analog sensors. By the time we complete this month's experiments, we will have covered the basic techniques that can be used to send a PICAXE ADC reading to the Pi, from any analog sensor we choose. So, let's get started!

MCP9700A vs. DS18B20

We're using the 9700A primarily because it provides the opportunity to experiment with sending analog values from the PICAXE to the Pi. However, if your main interest is in the temperature measurement itself, there are three good reasons to choose the MCP9700A rather than the DS18B20:

1. The DS18B20 is at least 10 times more expensive than the 9700A.
2. A PICAXE processor takes as long as 750 mS to fetch the digital temperature data from a DS18B20, but it only takes about 1.25 mS to read an analog input. In some projects, this huge difference really isn't very significant, but we ultimately want to program the Pi to interrupt the PICAXE whenever it (Pi) wants updated data.

As you may remember, the PICAXE checks for an interrupt condition after the completion of every instruction (and frequently as *wait* or *pause* commands are executed).

Consequently, if the PICAXE happens to be in the process of reading data from a 18B20 sensor when an interrupt occurs, it could take as long as 750 mS for it to respond, which would certainly complicate the Pi programming.

3. Finally, since the DS18B20 is a digital device, the Pi doesn't need a PICAXE to interface with it; it can do so all by itself! If you're interested in that approach, you can search for "Raspberry Pi DS18B20" (without the quotes) — you will find many relevant projects. (Also, you might want to check out #11 in the [Adafruit.com](http://adafruit.com)

Raspberry Pi Tutorials.)

We're going to conduct three different experiments this month, and all of them can be carried out with the same hardware setup, so let's begin with that. **Figure 1** presents the schematic of the circuit that we'll use for our experiments.

The first thing to note is that I haven't included the PICAXE programming connections to the 08M2 pins C.0 (*SerOut*) and C.5 (*SerIn*) but, of course, they are required!

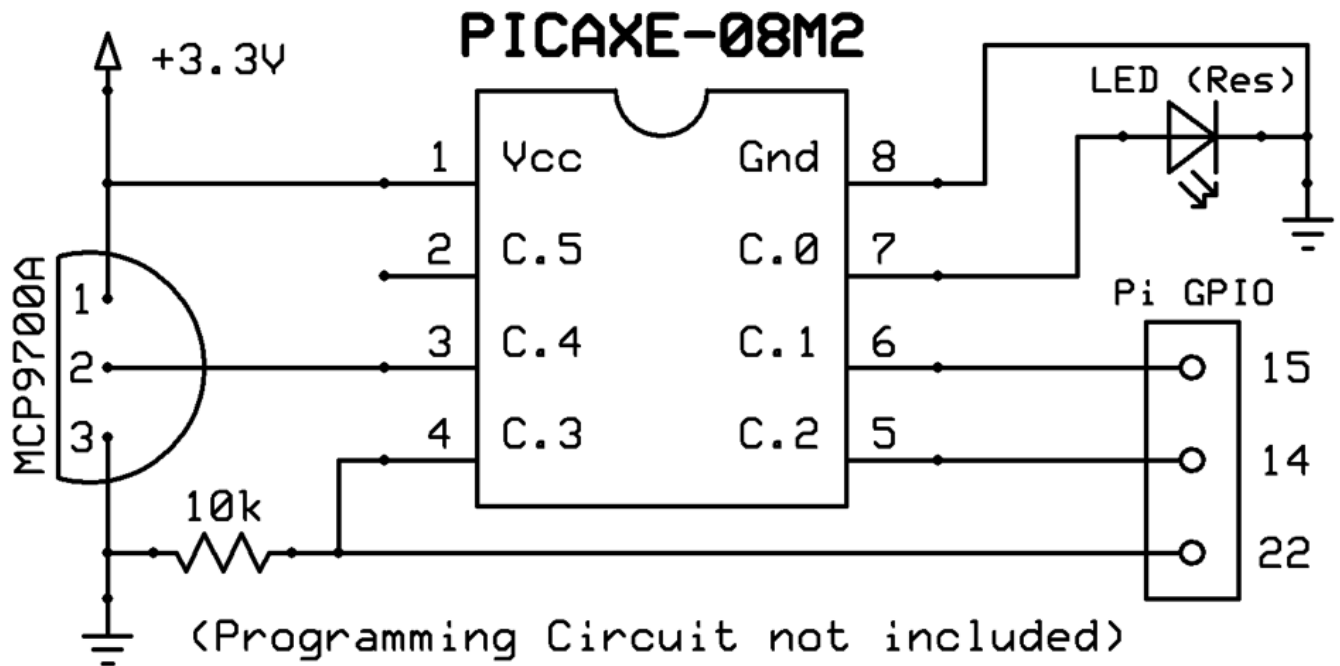
Also note that $V_{cc} = 3.3V$, which is the supply voltage for all our PICAXE-Pi projects. (In case you're wondering, the 9700A's supply range is 2.3V to 5.5V, so it will work fine in our 3.3V circuit.)

Finally, we won't be using the connection between GPIO 14 (TxD) on the Pi and pin C.2 on the 08M2. I only included it in case you (or I) want to conduct additional experiments with sending serial data from the Pi to the PICAXE.

Figure 2 is a photo of my breadboard setup for this month's experiments. As you can see, I'm again using the stripboard interface circuit that we constructed in our first PICAXE-Pi article (August 2013).

Of course, you can use any hardware setup you prefer — just make sure that the 08M2 is powered at 3.3V to match the levels on the Pi's GPIO pins, and that there is a series resistor in the I/O connections that we are using. (The stripboard interface circuit includes 470 Ω resistors in each GPIO line.)

If you are using the stripboard circuit, you may want to refer to **Figure 3** which presents the pinout of the GPIO header on the stripboard circuit.



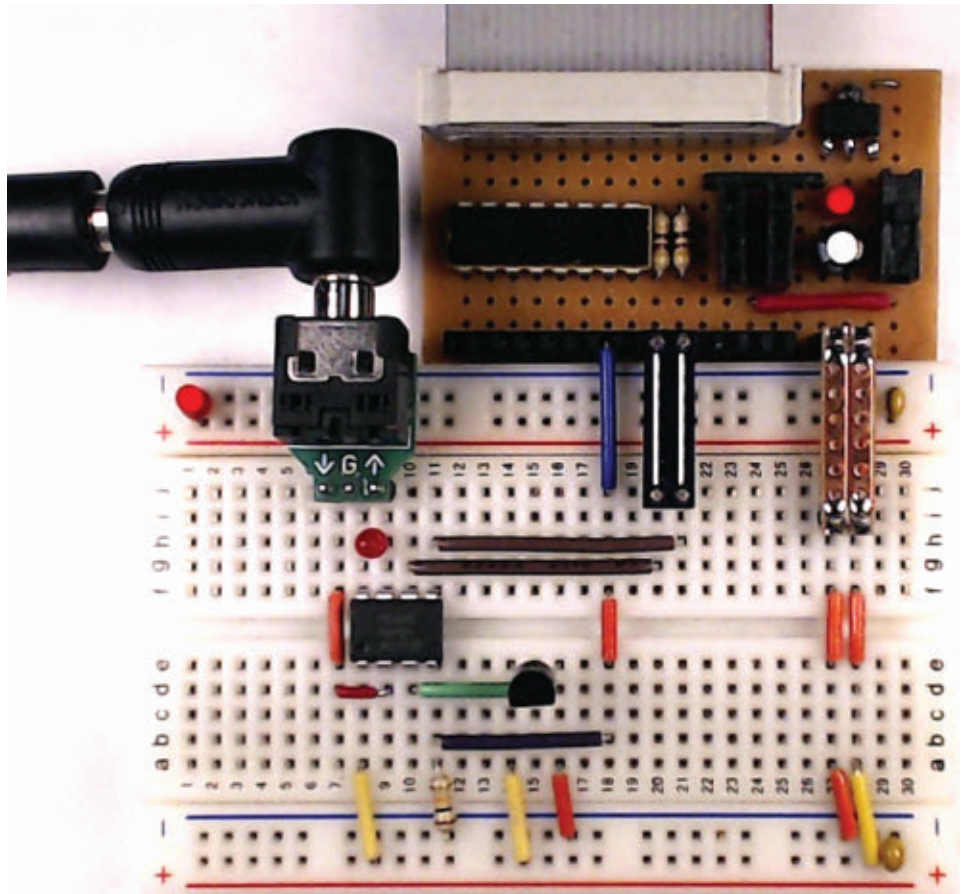
■ FIGURE 1. Schematic diagram for all experiments.

Experiment 1: Testing the MCP9700 Hardware Setup

We'll be using a total of six programs this month (three for the PICAXE and three for the Pi). Before reading any further, you may want to download the zipped file at the article link that contains the six programs.

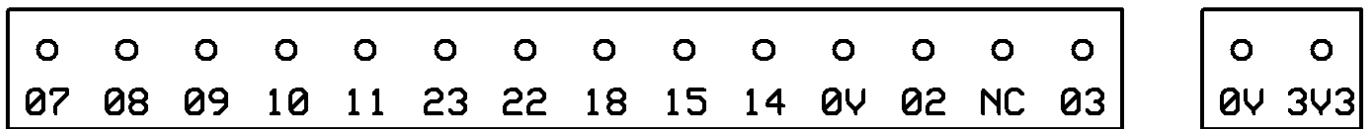
For our first experiment, we only need a PICAXE program (*temp9700test.bas*). When you have unzipped this month's program files, open *temp9700test.bas* in the PICAXE Editor and read through it. If it looks familiar, that's because it's essentially the same as the *Temp9700A.bas* program that we used in the December 2012 Primer.

The main difference is that, this time, we're displaying all three results (*ADCval*, *tempC*, and *tempF*) in the terminal window, rather than just the Fahrenheit temperature reading. (Also, we're not powering the 9700A from a PICAXE output pin this time, but you could do that if you prefer.)



■ FIGURE 2. Breadboard setup for all experiments.

Pi Breadboard Connector



■ FIGURE 3. Pinout of Pi breadboard connector.

The main purpose of our first experiment is just to test the hardware setup, so we don't need to get into the details of the program. (If you're interested, you may want to reread the December 2012 column.) At this point, just download the *temp9700test.bas* program to your breadboard setup.

You should see the three values (*ADCval*, *tempC*, and *tempF*) updating every five seconds in the terminal window. (Don't forget, even though there's a total of 10 seconds wait time in the main loop, we're running at 8 MHz, so the actual loop time is about five seconds.)

If you gently squeeze the 9700A between two fingers, you should also see the readings increase a bit, and slowly decrease when you remove your fingers from the sensor. When you're sure your hardware setup is functioning correctly, we're ready to move on to our next experiment, which will send the ADC data from the 08M2 to the Pi. However, before we do that, let's take a little break and play with a few of the features of the Python *print()* function.

Playing with Print()

The remainder of our experiments this month will all include the use of the Python3 *print()* function, so that we can visually check that the Pi is receiving the correct data from the 08M2. The *print()* function includes many advanced features that make it suitable for printing various reports that require precise formatting. We definitely won't be covering the majority of these features, but there are a couple of them we do need to

discuss, and I think it will be easier if we do that before we actually tackle the remaining experiments this month.

It may be tempting to compare the *print()* function with serial output such as the PICAXE *serout* and *sertxd* statements. However, there are important differences between the two operations. Serial output is frequently used to send data from one processor to another. In that situation, we only want the data transmitted (without any additional characters, such as spaces). In PICAXE BASIC, if we write *serout txPin, T9600_8, (hiByte,loByte)*, we expect to be sending exactly two bytes, and that's — in fact — what happens.

However, if we're transmitting something that's going to be read by a human (e.g., a *sertxd* transmission to the terminal window), we need to insert spaces (and punctuation characters) to make the output readable. In that case, we need to write something like *sertxd (tempC," ",tempF)* so that the two values are separated by a single space, and therefore easily readable.

On the other hand, the output of a Python3 *print()* function is almost always intended to be read by a human. So, by default, Python3 automatically includes a blank space between each argument in the *print()* function. In addition, it also prints a "new line" character at the end of the printed line, so that whatever we print subsequently appears on a separate line.

In order to clarify this, let's take a look at a few examples. Open the *printPlay.py* program in *idle3*. You will see two variable definitions (*tempC*

and *tempF*) and a numbered series of *print()* functions — all of which have been commented out. Each example includes a final "empty" print function, which is one way we can print a blank line between each example. As you read each of the following explanations, un-comment the corresponding functions in *printPlay.py*, and run the program to see the printed result:

1. The output is printed on one line with a space between each argument. However, sometimes we don't want a space between arguments. For example, the standard degree symbol is character 176 in the extended ASCII table. If we wanted to get fancy and use the symbol rather than the word "degrees," we might try #2.

2. Of course, the phrase is again printed on one line, with a space between each argument. That's definitely not what we want. There shouldn't be a space between the number and the degree symbol, so we need a way to override the default behavior of the *print()* function. Actually, Python3 provides multiple ways to accomplish our goal, but I'm just going to mention the two solutions that I have found to be the simplest to understand and use.

3. First, the *print()* function includes two optional arguments: *sep* (the "separator" character) and *end* (the "end" character). If we omit either or both of these arguments (which is what we have done so far), *sep* defaults to a "space" character, and *end* defaults to the "newline" character.

4. If we include *sep* and/or *end*,

we can set each one equal to any character or string that we want. Here, we just redefine *sep*. This example may seem silly, but it clearly demonstrates what happens when we include an optional definition for the *sep* argument.

It also should give you an idea of the formatting that's possible in Python3 printing. (We haven't yet solved our degree symbol problem, but we will before we're done!)

5. Here's another attempt to remove the space between the temperature and the degree symbol. As you can see, it also doesn't work!

6. This one does the job. We just needed to manually include spaces where we want them to appear.

7. Sometimes, a *print()* function is too long to conveniently type on one line. Here, we solve that problem by defining the end argument as the empty string. This example also demonstrates how we can switch between single quotes and double quotes, so that we can print an apostrophe in the text.

8. I mentioned earlier that I like two different Python solutions to our degree symbol problem. The second approach involves something called "string concatenation." In case you're not familiar with that term, it's simply a way of combining two or more strings (or characters) to make a single longer string.

In Python, the "+" symbol is used for two different purposes: addition and concatenation. If we place a + symbol between two numbers, Python adds them (duh!), but if we place the same symbol between two strings (or a string and a character, or two characters), Python concatenates them. The examples in this section are somewhat trivial, but they do demonstrate how concatenation works.

9. Here, we use concatenation to solve our degree symbol problem. Note that we have to convert the two temps from numbers to strings for this to work. Also, note how we can define *end* as the empty string to

get everything to print on one line.

Now that we've covered a few details of the Python3 *print()* function, we can move on to our remaining experiments. As you encounter the various *print()* functions in the upcoming Python programs, you may want to refer back to the above examples for clarification. Better yet, you may want to try a few of your own!

Experiment 2: Simple Serial Transmission from PICAXE to Pi

If you look again at the listing for the *temp9700test.bas* program, you can see that a significant portion of the code is devoted to the conversion of the raw data; first to Centigrade, and then to Fahrenheit. The conversion code is the most complicated portion of the program, and it results from the PICAXE's inability to perform computations on negative integers, fractions, or decimal numbers. Fortunately, we're about to get some mathematical help.

Since the PICAXE is helping the Pi to deal with analog inputs, the Pi is going to return the favor, and help the PICAXE with the math! In other words, we're going to send just the raw data to the Pi and let it do the math.

First, take a look at the listing for the PICAXE program for Experiment 2 (*tempSerialToPi.bas*). As you can see, it's much simpler than the program we used in Experiment 1. The subroutines have been completely eliminated; the main *do/loop* is less complicated; and there are fewer variables.

However, there is one small complication. We need to include two new variables: *hiByte* and *loByte*. This is necessary because all serial communication is byte-based. In other words, we can't simply send the *ADCval* to the Pi because that's a 16-bit word value; we need to send the high byte and low byte of that value separately.

Since *w0* is comprised of *b1* (high byte) and *b0* (low byte), we've defined suitable names for the two bytes so that we can serially transmit them to the Pi.

Also, note that it may look like we're sending a single 16-bit value in the *sertxd* statement, but the inclusion of the "#" symbol in a *sertxd* or *serout* statement actually results in a 16-bit value being transmitted as 16 separate bytes. We could have used the same approach, but it's much easier to just send two bytes and let the Pi reconstruct the 16-bit value of *ADCval*.

Now, let's turn our attention to *tempSerialFromAx.py* (which is the Pi program for Experiment 2) and discuss the *try* portion of the main *while loop*. In the program listing, some of the program comments are numbered. The following comments elaborate on the corresponding program comments:

1. As we discussed in the previous Primer, the *pySerial read()* function reads a single byte, and it's a blocking function by default which means the program will remain at that point until a serial data byte is received. Of course, in a real world program, that would be unacceptable — we don't want our program to "hang" forever!

However, we're just using this approach in our first example to keep things simple. Before we're finished this month, we'll implement one way to avoid that problem.

2. Here, we reconstruct the 16-bit value of *ADCval*. As you can see, it's a simple task.

3. In the December 2012 column, we used this equation to have a PICAXE processor convert the value of *ADCval* to Centigrade: $\text{tempC} = (2 * \text{ADCval} - 500) / 10$. This time, however, the Pi is doing the math.

Negative numbers and/or decimals are no problem for the Pi, so we can further simplify the above equation to $\text{tempC} = 0.2 * \text{ADCval} - 50$;

PortC Pins Usable for Interrupts

Processor	C.7	C.6	C.5	C.4	C.3	C.2	C.1	C.0
08M2				✓	✓	✓	✓	
14M2						✓	✓	✓
18M2	✓	✓	✓	✓		✓	✓	✓
20M2			✓	✓	✓	✓	✓	
20X2			✓	✓	✓	✓	✓	

■ FIGURE 4. Port C pins for PICAXE interrupts.

the Pi can do the entire computation in a single step.

4. Python's `round()` function takes two arguments: the value that we want to round; and the number of decimal places we want to use.

5. In this statement, we're converting the temperature reading from C to F by doing the same two computations we just carried out to convert `ADCval` to Centigrade. However, this time, we're doing them both in a single statement.

Python computes the value of `tempF` using the standard C to F conversion formula, and then rounds the result to one decimal place — all in one line of code.

6. These three `print()` functions use some of the features we demonstrated earlier with the `printPlay.py` program. The second and third `print()` functions demonstrate the two methods we discussed for formatting the data.

You may want to refer back to that discussion to see if you can predict how the printed output will be formatted.

When you're ready to carry out Experiment 2, run the Python program first. The blocking `read()` function will cause it to wait until the PICAXE program begins to send the serial data.

Experiment 3: An Interrupt-Triggered Serial Transmission

In this experiment, we're putting the Pi in charge of when a serial transmission occurs. The Pi will issue an interrupt signal and the 08M2 will respond by serially sending the raw temperature data (`ADCval`) to the Pi. We're again transmitting two data bytes (`hiByte` and `loByte`) to the Pi, but the same approach can easily be used for any amount of data.

In other words, it would be a simple matter to serially connect a larger processor (such as the PICAXE-20M2) to the Pi and have the 20M2 monitor several sensors, then send all the updated data whenever the Pi issues an interrupt. Also, we're going to unblock the Pi's serial reception so that we don't run the risk of the Pi hanging because there has been a glitch in a serial transmission.

All current PICAXE processors support some form of interrupt capability, but we're going to limit our discussion to the M2-class processors and the 20X2. If you're interested in the more advanced features of the 20X2 and 40X2 processors, you may want to read the relevant documentation on the `setint` and `setintflags` commands in

Section 2 of the PICAXE manual.

Also, we discussed interrupts way back in the February 2009 Primer, so I won't repeat all the details here. If you want more information than we're about to discuss, you may want to reread the 2009 article, and the *setint* documentation in the manual.

Let's begin with a brief explanation of how an interrupt functions. First, we need to include a `setint` command at the beginning of our program to specify which pin(s) we want to use for the interrupt, and whether we want the interrupt to be triggered by a high or low pulse. (We'll see exactly how to do that shortly.)

Secondly, we also need to include an interrupt subroutine which must begin with the label `interrupt:` and end with a `return` statement. The `interrupt` code specifies the action we want to take place in response to the interrupt signal. (In this case, the 08M2 will send the values of `hiByte` and `loByte` to the Pi.)

When we run the PICAXE program, after each program line is executed (and continuously during any `pause` or `wait` command), the PICAXE compiler checks to see whether an interrupt condition exists. If it does, the `interrupt` subroutine is immediately executed, and then program execution continues at the

next instruction that was about to be executed when the interrupt occurred.

Now, let's turn our attention to the *setint* command, and how we specify the input condition(s) and the pin(s) that we want to use for the interrupt. The complete syntax is *setint input, mask*, where *input* defines the desired input condition(s) (high/low), and *mask* defines which pin(s) are to be checked to see if an interrupt should occur.

Since the interrupt we need is a simple one (i.e., one pin and one condition), let's use that as an example to clarify the use of the *setint* command. First, on all current PICAXE M2-class processors (and the 20X2), only port C pins can be used to trigger an interrupt. In addition, each processor is limited to specific port C pins. **Figure 4** presents the available pins for each processor. In the present experiment, we'll use C.3 on the 08M2 because it's fixed as an input, and it can't be used for an ADC reading.

When I first started working on this experiment, I assumed it would make sense to use the built-in pull-up resistor on pin C.3 to hold the input line in a high state, and have the Pi interrupt the PICAXE by pulling C.3 low. However, that didn't work very well. When a program is first run on the Pi, all the input pins are set low (which is the same safety precaution that PICAXE uses), so a "false" interrupt is generated every time a Pi program starts to run.

As a result, I switched to the other option (pin C.3 normally low, with a high interrupt pulse). We could use the internal pull-down resistor on the Pi to hold GPIO 22 low, but I decided to use an external 10K pull-down resistor (see the schematic in **Figure 1**), just to remind myself that the interrupt line is being held low.

Now, let's examine the *setint* command we need to implement the interrupt (*setint %00001000, %00001000*). We'll start with the second parameter (*mask*) because it

makes things easier to understand. As mentioned, the *mask* specifies which pins are to be checked. The fact that there is only one "1" in the mask (in the bit 3 position) indicates that we are only interested in checking the state of pin C.3.

In the first parameter (*input*), the 1 in the bit 3 position indicates that

we want the interrupt to occur when pin C.3 is pulled high. This syntax may seem unnecessarily complicated, but that's because we're implementing a simple one-pin interrupt.

If, for some reason, we wanted to implement an interrupt that should be triggered whenever pin C.3 is high and pin C.2 is low, we would write

ALL ELECTRONICS

www.all-electronics.com

Order Toll Free 1-800-826-5432

1.5 AMP SOLID STATE RELAY

CP Clare OptoFILM™ #OFB2402
Control Voltage: 9-16 Vdc
Output Rating: 1.5A / 240 AC
Optically isolated, pc mount solid state relay. 22.2 x 12.7 x 6.4mm. CSA, VDE.
CAT# SRLY-2402

\$2.20 each
10 for \$2.00 each

PRECISION 10K POT, USED

Spectrol/ Vishay Model 157
Aluminum housing. Ceramic sealed back. 10K Ohm linear taper. 1 Watt. 1/4" flatted stainless steel shaft. 3/8-32 threaded bushing mount. Removed from equipment with short wires soldered to turret terminals. May have writing or test marks.
CAT# PPT-10KLU

\$1.00 each

6" DC μ A METER, USED

125 μ A DC = FS.
High-quality 6" wide DC panel meter. Solid plexiglass enclosure, 6" wide x 4.25" oval. Removed from equipment.
CAT# PMD-125U

\$5.00 each

47UF 250V 105°C

Nichicon UCS2E470MHD1CT.
47UF, 250V radial. 105°C. 12.5mm dia. x 20mm long. 3.8mm (0.15") cut leads.
CAT# 47R250

45¢ each
10 for 40¢ each • 100 for 30¢ each

25MM STEPPER MOTOR

25mm (1") dia. x 12mm stepper motor. 6 leads, 3" long. Mounting holes on 35mm centers. 3.5mm dia. threaded shaft.
CAT# SMT-136

\$2.00 each

NEODYMIUM MAGNET, 28 X 16 X 4.5MM

Very strong for its size, 18-22K gauss. Irregularly shaped. 1.10" x 0.63" x 0.19".
CAT# MAG-148

Reduced Price! \$1.25 each
10 for \$10.00

2-CONDUCTOR AC CORD

6' black 18AWG/2 SVT power cord. Molded polarized 2-prong plug. Molded strain relief. Pigtail leads with 0.25" QC fully-insulated connectors. UL, CSA.
CAT# LCAC-419

\$2.00 each
10 for \$1.85 each • 150 for \$1.75 each

12V BLUE LED, 5MM

Bright 5mm blue LED (milky-white in off-state) with built-in resistor for 12 Vdc. No external resistor required. Works well on 4-12Vdc. Dimmer at lower voltages. Diffused. 35° viewing angle.
CAT# LED-12BW

55¢ each
100 for 45¢ each
1000 for 35¢ each

`setint %00001000, %00001100`, so the same syntax works for more complicated interrupts as well.

One final point: Whenever a program jumps to its *interrupt* routine, the compiler immediately disables the interrupt. If it didn't, the *interrupt* would most likely interrupt itself multiple times! For example, if the Pi uses a 1 mS high pulse to trigger the interrupt, pin C.3 could still be high when the PICAXE program enters the *interrupt* routine which, of course, would again trigger an interrupt; immediately disabling the *interrupt* avoids that problem.

However, it also means that we need to include another `setint` command (identical to the one at the beginning of the program) just before the program returns from the *interrupt* routine, so that the interrupt is re-enabled.

Now that we've covered the

basics of interrupts, we're ready to take a look at the two programs we'll be using in this experiment. First, open the PICAXE program (`sendReport.bas`). As you can see, it's almost identical to the program we used in Experiment 2, so it requires little explanation.

The only significant difference is that the serial transmission has been moved from the main *do/loop* to the *interrupt* routine. Also, note how we re-enable the interrupt before returning from the *interrupt* routine.

The companion program for the Pi (`requestReport.py`) is a little more involved, so there are a few details that may require clarification. Open the program in the *idle3* editor, and take a look at the main portion of the code. In the first program line, the serial port is opened, but this time we're including the optional *timeout* argument and setting its value to one

second. As a result, whenever a `ser.read()` statement is executed, if a data byte is not received within one second, the program will move on.

At the end of the *try* block in the main *while* loop, the *for* loop also requires a brief explanation. When I first wrote the program, I simply used a `sleep(10)` statement at this point because I wanted the program to check for new data every 10 seconds. However, it appears that Python can't respond to the *ctrl-c* keyboard interrupt when the program is sleeping. If I pressed *ctrl-c*, it took as long as 10 seconds for the program to terminate; the *for* loop solved that problem.

Now, let's turn our attention to the `getData()` function which does most of the work in the program. The following comments elaborate on the correspondingly numbered program comments in the `getData()` function:



ROS-Training


Diagram illustrating the ROS-Training components: CONTROL, PERCEPTION, LOCATION, and INTELLIGENCE, arranged around a central grid icon.

SPECIAL OFFER: 20% OFF YOUR FIRST CLASS!

To reserve your discount sign up at www.ros-training.com/nuts_volts

LEARN • BUILD • SHARE

www.ros-training.com



Preview the lessons at: learn.parallax.com/ActivityBot

Great for STEM learning!

ActivityBot

ROBOT KIT \$199

Learn to program in C on Mac or PC. Online tutorials are suited for the self-guided learner yet easily expand to accommodate a classroom or club setting.

PARALLAX

www.parallax.com

1. The `pySerial flushInput()` function empties any data that happens to be in the Pi's serial buffer to make sure that the two `ser.read()` functions fetch the most recent data. It may not be absolutely necessary to include this statement, but it does avoid the possibility of extraneous data accumulating in the buffer if a program is run for a considerable period of time.

2. As we already know, both the `ser.read()` statements in this program are non-blocking. If a data byte hasn't been received in the specified time, a timeout occurs and the program moves on. However, a timeout also produces a Python exception. If our program didn't handle this exception, the program would automatically terminate immediately with an error message. In this case, the message would be something like "`ord()` expected a character, but string of

`length 0 found`" because no data was received. One way to handle the exception (and avoid the program termination) is to place the `ser.read()` statement within a `try/except` block as we have done here.

3. Just before the `try/except` block, we initialized a Boolean "valid data" flag (`valid`) as `True`. If an exception occurs, the `except` block is executed, so here we update `valid` as `False`.

4. Finally, in the `if/else` code, we first check to see if the data is valid. If it is, we carry out the necessary computations and return the value of `tempF`. If it's not valid, the `else` code executes and we return the value of 999 which indicates that the data is not valid.

Let's run the two programs and see what happens. First, run the `requestReport.py` program. (Don't forget: You need to run as `root`,

because we're using `RPi.GPIO`.) You should see "`Warning: The data is invalid!`" printed in the `idle3` editor every 10 seconds. That's because the 08M2 isn't sending any data yet, so every `ser.read()` statement produces a timeout. Next, run the `sendReport.bas` program; you should see a (nicely formatted!) valid temperature reading every 10 seconds.

It seems like we've invested a huge amount of effort just to know what the temperature is. However, the real point of all this work is that we now have a programming template that enables us to use any PICAXE processor to collect a variety of data (analog and/or digital), and send it on to the Pi whenever it asks for it. Since the Pi is easily connected to the Web, this opens up a whole new world of PICAXE programming possibilities. Think about that and have fun! **NV**



AP CIRCUITS
PCB Fabrication Since 1984

As low as...

\$9.95
each!

Two Boards
Two Layers
Two Masks
One Legend

Unmasked boards ship next day!

www.apcircuits.com



RF Specialists

Affordable RF Modules

Introducing the LMX-ISM-242-SR and LMX-ISM-242-LR



Data Loggers

Stand Alone and Wireless Mesh Networking Logger



Industrial Bluetooth

OEM, Modules, Wireless Device Servers, RS-232 Long range options, low cost



ZigBee Pro

OEM Modules and USB ZigBee Sticks, Mesh Networks



Low Power, High Performance Semiconductors

Excellent Sensitivity, Very Efficient Power Amplifiers

Narrow Band, General Purpose, TSSOP, RF Microcontrollers
Typical Sensitivities are in the range of -110dBm to -126 dBm for 1.2 kbps

RF Design Services

Prepared to work with your in-house engineers, or support your RF project from initial design to implementation.

Industrial • Military • Space • Medical • Smart Grid Metering • SCADA • Lighting Control



LEMONS
INTERNATIONAL

www.lemosint.com
866.345.3667
sales@lemosint.com

"Making your RF ideas into profitable products."

NEW PRODUCTS

■ HARDWARE
■ SOFTWARE
■ GADGETS
■ TOOLS

PCB ASSEMBLY KIT

Beta LAYOUT Ltd. has developed a cost-effective printed circuit board (PCB) assembly kit enabling PCB designers, universities, labs, and home hobbyists to hand assemble SMD components utilizing a toaster oven. For less than \$406, any small space can become a PCB assembly reflow workstation.

To control and establish uniquely designed solder profiles is the US generation Reflow-Controller (V3) Pro, which can be used in conjunction with a basic toaster oven to create the temperature profile needed to reflow components and create professional assembled boards. Other aspects include:

- Learning the profile and having five pre-selected profiles digitally displayed.
- With the press of a button, a profile can be repeated time and time again promoting component soldering consistency.
- Base values can be easily adjusted for various reflow requirements.

The Reflow-Controller (V3) allows a desired pre-heat phase to solder paste melting phase, avoiding mechanical stresses on the PCB and components.



After preliminary heating, the temperature is increased to just under the soldering temperature. This allows the volatile components of the flux to escape, eliminating blistering. As the heat increases, the flux melts forming a connection between the components and PCB. The soldering temperature is accurately adhered to during the soldering phase, so that the PCB is not damaged due to overheating.

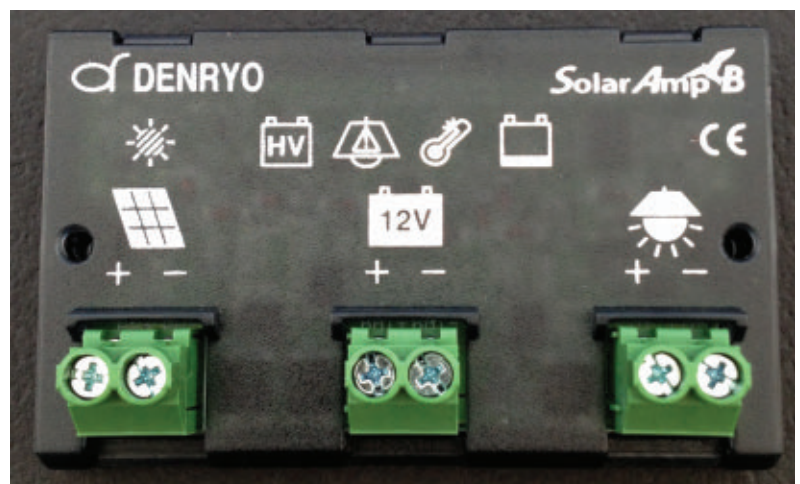
To complete the kit, users can take advantage of free laser SMD stencils offered by Beta LAYOUT. To secure the stencil and enable precise solder paste applications, a magnetic workbench is also available. The final result is components are easily assembled to the paste and reflowed in the oven.

For more information, contact:
Beta LAYOUT, LTD.
Web: www.pcb-pool.com

SOLAR PANEL BATTERY CHARGE CONTROLLER

J2 LED Lighting, LLC introduces the Denryo SA-BA10 solar charge controller intended for solar-powered LED lighting applications. The SA-BA10 is a Japanese engineered controller with sophisticated battery charge control ability in a simple to use lightweight and robust module.

The SA-BA10 is rated at 10 amps maximum solar panel current for a 12 volt DC system. The controller can typically support up to a 170 watt



solar panel or two 85 watt panels parallel wired, provided either configuration does not exceed 10 amps total per panel ratings for maximum current.

The controller has five LED status indicator lights:

1) charge indicator (green); 2) battery over voltage (red); 3) current over load (red); 4) controller high temperature (yellow); and 5) battery low voltage (yellow). The status indicator lights are under a translucent smoke black high strength polycarbonate plastic cover with silkscreen icons for the given status functions.

The SA-BA10 accepts 12-18 AWG wire into its terminal blocks with screw type clamps for power interconnects. Circuit protection should be set to a maximum of 10 amps with an appropriate fuse on the battery, solar panel, and load. The controller is designed for charging lead-acid batteries; the controller provides a battery bulk charge voltage of 14.4 and a float charge of 13.7 volts.

The controller has three battery over-discharge voltage points. The first is a warning when the battery is at 11.8 volts, indicated by the battery status indicator light flashing. The second is when the battery is at 11.5 volts for battery disconnect, indicated by the battery light on steady. The third is when the battery is at 12.5 volts, indicated by the light turning off. The load is then automatically reconnected.

The controller's battery over-discharge protection system is designed to provide for long battery service life. An important factor for good battery life is limiting battery discharge to the typical voltage point at which the usable amp hour (Ah) capacity of the battery has been reached. The solar controller's idle state current is very low at 2 ma typical; this reduces battery drain when there is no panel voltage to the unit and no load is being drawn.

The controller is intended for applications in which the controller is protected from direct exposure to moisture and water. The controller is designed for operating in ambient temperatures -20°C (-4 °F) to 60°C (140°F). For marine, dock side, or other heavy water and moisture exposure environments, the controller can be mounted in an appropriate NEMA or IP rated enclosure. Each unit ships with a user's manual.

The SA-BA10 is suitable for many applications including the following:

- LED lighting of out buildings and sheds.
- Portable solar panel LED lighting systems.
- Charging of lead-acid battery packs.
- Dock side and marine lights.
- Solar-powered signage LED lights.

Pricing is as follows:

- 1) \$36.99 for single piece to four pcs
- 2) \$33.99 for 5-24 pcs
- 3) \$31.49 at 25 pcs and over

SOLAR AMP MINI

J 2 LED Lighting is also introducing the Denryo SA-MN05-8 Solar Amp Mini solar PV panel charge controller intended for solar-powered LED lighting control applications. The SA-MN05-8 is also a Japanese engineered controller with sophisticated battery charge and lighting control ability in a small robust module.

The SA-MN05-8 is rated at 8.5 amps maximum solar panel current for a 12 volt DC system. The controller can typically support up to a 150 watt solar panel or two 75 watt panels parallel wired, provided either configuration does not exceed 8.5 amps total per panel ratings for maximum current.

The controller has four operating mode functions for LED lighting control: 1) night light off = load always on; 2) dusk to dawn = lighting load output on at night; 3) normal timer = night on time of 6, 8, 10, 12, or 14 hours; and 4) rate timer = light load on for a percentage of night time, 40%, 50%, 70%, or 80%. Functions are programmed from the user interface front panel and the unit's memory will store a program up to two years with no power to the unit. Light level sensing for lighting control is via solar panel voltage levels monitored by the controller.

The SA-MN05-8 accepts 16-22 AWG wire into its terminal blocks with screw type clamps for power interconnects. Circuit protection should be set to a maximum of 10 amps with an appropriate fuse on the battery, solar panel, and load. The controller supports four different battery types of lead-acid chemistry: 1) sealed (SLA); 2) absorbed glass mat (AGM); 3) gel cell; and 4) flooded (wet cell).

The controller has battery low voltage disconnect (LVD) capability when the battery is at 11.5 volts. Battery reconnect occurs at 12.5 volts. The controller's battery over-discharge protection system is designed to provide long battery service life. An important factor for good battery life is limiting battery discharge to the typical voltage point at which the usable Ah capacity of the battery has been reached. The solar controller's idle state current is very low at 1 ma typical. This reduces battery drain when there is no panel voltage to the unit and no load is being drawn.

The controller is intended for applications in which the controller is protected from direct exposure to moisture and water. The controller is designed for



operating in ambient temperatures -20°C (-4 °F) to 60°C (140°F). For marine, dock side, or other heavy water and moisture exposure environments, the controller can be mounted in an appropriate NEMA or IP rated enclosure. Each unit ships with a user's manual. The SA-MN05-8 is suitable for many applications including the following:

- Parking lot solar-powered LED lights.
- Dock side and marine LED lights.
- Gazebo solar LED lighting.
- Golf course and pool side solar-powered LED lighting.

Pricing is as follows:

- 1) \$56.99 for single to four pcs
- 2) \$53.99 for 5-24 pcs
- 3) \$51.49 for 25 pcs and over

For more information, contact:
J2 LED Lighting, LLC
 Web: www.j2ledlighting.com

ICONSTRUX.COM LAUNCHES

In 2003, computer scientist and author, Andre' LaMothe designed and delivered the do-it-yourself game console called the "XGameStation Micro Edition." As the first product of its kind, a whole site was dedicated to support and sell the product at www.xgamestation.com. Over the years, the XGameStation Micro was followed up with numerous other educational and edutainment based embedded gaming systems used by individuals, as well as many college level embedded systems courses.



In 2013, the creators of the XGameStation brand were frustrated with the retail websites available online for embedded systems and electronics enthusiasts. So, after a year of labor, a new site has emerged: iCOnstruX.com.

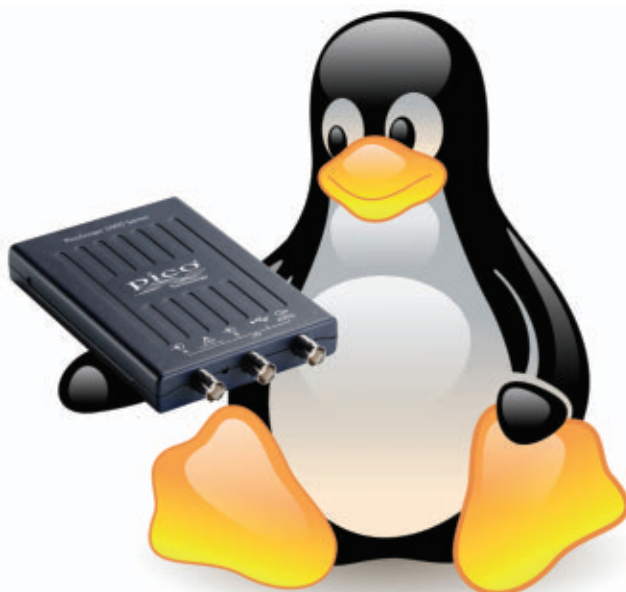
The site is targeted to real hackers and makers — people that build stuff — so there are a lot of embedded systems, gizmos, and gadgets, as well as complete kits, toys, and beginner products.

One of the unique things about iCOnstruX.com is the community which is helping Indie hardware developers by leveraging their vast experience in product development and manufacturing. So, iCOnstruX.com has programs where Indie developers can get help, consulting, and complete manufacturing support for new products they want to try and develop.

For more information, contact:
iCOnstruX.com
 Web: www.iCOnstruX.com

SOFTWARE TURNS PCs INTO OSCILLOSCOPES

Saelig Company, Inc., announces the availability of Linux-based software for PicoScope PC-based oscilloscopes. PicoScope 6 For Linux is a powerful



application that allows a PC to be connected to a PicoScope USB adapter to produce a high-powered oscilloscope, FFT spectrum analyzer, and data acquisition device. With built-in buffering in the PicoScope adapter, the PC's display is updated frequently and smoothly even when set on long timebases. Previously only available as Windows-based software, PicoScope 6 For Linux includes

Recycling & Remarketing High Technology

WEIRDSTUFF®

WAREHOUSE

Software, Computers, Electronics, Equipment, Doohickies

WE BUY/SELL EXCESS & OBSOLETE INVENTORIES

FREE COMPUTER AND ELECTRONIC RECYCLING

GIANT 10,000 SQ. FT. AS-IS SECTION

384 W. Caribbean Dr.
Sunnyvale, CA 94089
 Mon-Sat: 9:30-6:00 Sun: 11:00-5:00
(408) 743-5650 Store x324
WWW.WEIRDSTUFF.COM

www.Primecell.com

Battery rebuilding service

Dead Batteries ? Don't toss them.
 Send them to us - our rebuilds are better than original specifications.

Tools

Electronics

Radios

Visit www.primecell.com for important details
 24 Hr Secure recorder tel-fax (814) 623 7000
 Quotes email: info@primecell.com
 Cunard Assoc. Inc. 9343 US RT 220 Bedford PA 15522

QKITS LTD
sales@qkits.com
1 888 GO 4 KITS

Speed Controllers
 Timers and Controllers
 Soldering Supplies
 Audio Kits
 LED Kits

Visit us at:
www.qkits.com

In the Nuts & Volts Webstore NOW!

NUTS AND VOLTS

NV BOOK SPECIALS

Programming the Raspberry Pi
 Getting Started with Python

Raspberry Pi Three Book Combo

Raspberry Pi User Guide

Raspberry Pi Projects for the EVIL GENIUS

Only \$48.95
 Plus
 FREE Priority Mail Shipping
 US Only

To order call 800 783-4624 or visit:
<http://store.nutsvolts.com>
 Limited time offer

Redwood Electronics Corporation

industry-specific
 closed circuit
 video equipment

service-oriented
 consumer parts
 department

call: (800) 227-1768

Small Mechanical Components for

ROBOTICS

800-819-8900

SDPSI Visit sdp-si.com
 and Buy Online Today.

NK electronics

THE OFFICIAL ARDUINO STARTER KIT!

NKElectronics.com/starterkit
 Includes the Arduino Project Book (170 pages)

Purchase Orders are accepted from Educational Institutions,
 US Government and Research Centers

SLIDE MOUNT for MOBILE RADIOS
UNIVERSAL: FITS MOST 2 WAYS

Model ET-850 Universal Slide Mount. Standard mounting platform, specify radio make/model. Ends blind fumbling to release hidden fasteners. Easily transfer radios from vehicle to vehicle. Also use in the office, job trailer, marine too. Black powder coat finish. Volume prices.

(Ask about our maximum security Locking Radio Cage™ Mount)

E TIP, Inc., North Aurora, IL 60542 USA
 (800) 530-5064 FAX (630) 801-9569
www.etipinc.com | sales@etipinc.com

E TIP is a Veteran Owned Small Business

a wide range of standard oscilloscope features such as waveform display, spectrum display, interactive zoom, sophisticated triggering, automatic measurements, and signal generator control. Waveforms can be captured for off-line analysis or sharing with other users, they can be exported as text, CSV, and Mathworks MATLAB 4 formats.

Pico Technology's real time oscilloscopes are compact, economical USB adapters available with bandwidths up to 1 GHz, with up to four input channels, hardware vertical resolution to 16 bits, sampling rates up to 5 GS/s, buffer sizes up to 2 GSa, and built-in signal generators. Other features available on some models include flexible hardware resolution, switchable bandwidth limiters, switchable high-impedance and 50 ohm inputs,

and differential inputs. All of these adapters now run on Linux-based PicoScope 6 software.

The new PicoScope 6 For Linux software is packaged for easy installation on the following distributions:

- Debian 7.0 (wheezy) i386/amd64
- Ubuntu 12.xx/13.xx i386/amd64
- Any other Debian-based distribution with mono-runtime >= 2.10.8.1

PicoScope 6 For Linux is available free of charge.

For more information, contact:
Saelig Company, Inc.
 Web: www.saelig.com

Continued on page 80

BUILD THE BaTESLA COIL

By Matt Bates

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/april2014_Bates



I like to look at patents from the past, and often wonder what the inventor would have done with modern day hardware. I was recently looking at Tesla patents that describe the infamous Tesla coil and wondered how Mr. Tesla would have implemented his concept of wireless energy transmission if he had access to 21st century electronic parts like transistors and dielectrics. I have always wanted to build a Tesla coil, but have been put off by the tedium of tweaking spark gaps and dealing with dangerously high voltages. I decided to try building something loosely based on the original Tesla coil using much lower and safer input voltages that could at least be powerful enough to enjoy Tesla effects such as wirelessly lighting nearby CFLs (compact fluorescent lamps) and to study the concept. This article is a review of how I went about building my own version of a solid-state tabletop Tesla-like coil using common off-the-shelf parts. It won't exactly light a city, but it is a lot of fun to play with.

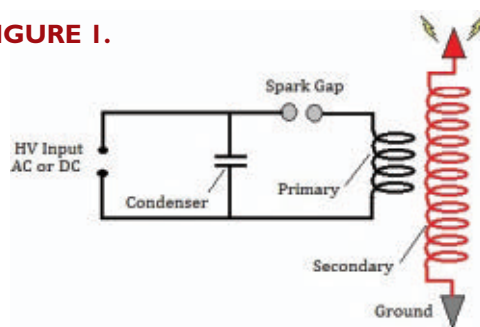
What Makes This Design So Different?

I am not going to try and cover a detailed explanation and theory behind this technology. Instead, I will ask that you visit any of the many Internet sites that cover the specifics. Suffice it to say that in a conventional Tesla coil, the primary and secondary inductors share the same axis and are located close to one another. In this manner, the magnetic field produced by one inductor can generate a current in the other.

The schematic in **Figure 1** shows the basic components of a Tesla coil. The primary oscillator (or tank circuit) consists of a flat spiral inductor with only a few turns, a capacitor, a voltage source to charge the capacitor, and a switch or spark gap to connect the capacitor to the inductor. The secondary oscillator contains a large tightly wound inductor with many turns, and a capacitor formed by the earth on one end and an output terminal (usually a sphere or toroid) on the other.

A high voltage power supply charges up a capacitor. When the capacitor reaches a high enough voltage, the spark gap fires. The spark gap is like a switch in that it conducts when the voltage gets high and turns off when

■ **FIGURE 1.**



the voltage gets low. When the spark gap fires, the energy stored up in the capacitor dumps into a 1:100 step-up transformer. The primary is about 10 turns of heavy wire. The secondary is about 1,000 turns of thin wire. With this ratio, if you feed in 10,000 volts, you get out 1,000,000 volts. It all happens at a rate of over 120 times per second, often generating multiple discharges in many directions.

The BaTESLA coil does not rely on a tank circuit for the oscillations and – perhaps best of all – it features auto-tuning. With my design, the PIC generates the frequency and applies it to the primary coil by way of an

ITEM	QTY	DESCRIPTION	SOURCE
SW1	1	Toggle Switch	RadioShack Catalog #275-602
SW2	1	Slide Switch	Jameco Catalog #109171
T1	1	25V Transformer	RadioShack Catalog #273-1366
Q1	1	TIP3055 NPN	RadioShack Catalog #276-2020
HS	1	TO-220 Heatsink	RadioShack Catalog #276-368
R1,2	2	120 ohm 1/4W	Digi-Key Catalog #120QBK-ND
R3	1	10K ohm one watt	Digi-Key Catalog #10KWCT-ND
R4	1	220 ohm 1/4W	Digi-Key Catalog #220QBK-ND
R5	1	1 megohm 1/4W	Digi-Key Catalog #1MAACT-ND
R6	1	10K ohm 1/4W	Digi-Key Catalog #A104668CT-ND
R7	1	22K ohm 1/4W	Digi-Key Catalog #22KAACT-ND
F1	1	0.5 amp Fast Blow	RadioShack Catalog #270-1056
FH	1	Fuse Holder	RadioShack Catalog #270-739
IC1	1	08M2 PICAXE	SparkFun.com #10803
ICS	1	Eight-pin IC Socket	RadioShack Catalog #276-1995
D2	1	3 mm Green LED	RadioShack Catalog #276-009
Lp	1	1,000 μ H Choke	Jameco Catalog #642927
BR1	1	Bridge Rectifier	RadioShack Catalog #276-268
BR2	1	Bridge Rectifier	RadioShack Catalog #276-1173
J1	1	Programming Jack	Jameco Catalog #1766180
Z1	1	4.7V Zener Diode	Jameco Catalog #178773
PCB	1	Gen Purpose PCB	Jameco Catalog #206587
C2	1	100 μ F 200V Cap	Digi-Key Catalog #338-2372-ND
C3	1	1 μ F 50V Electrolytic	Jameco Catalog #94161
C4	1	.01 μ F Capacitor	Jameco Catalog #25523
JCK	1	120V Power Jack	Digi-Key Catalog #486-2095-ND
PLG	1	120V Power Cord	Digi-Key Catalog #AE9906-ND
C1	1	1 μ F Metallized Polypropylene Film Capacitor (see text)	

Misc. Hardware:

1/8" Lexan material
200' 30 AWG Magnet Wire
L3 Coil Form 2-3" diameter non-metallic tube
L2 Coil Form 1.5" diameter non-metallic tube with caps
Threaded Standoffs with associated mach screws

**PARTS
LIST**

All of the designs produce copious amounts of RF energy which can wreak havoc with nearby unshielded

Overview of the Circuit

Constructing the Coils

The secondary coil is wound using 30 AWG wire. The choice of diameter for the coil will dictate how long a single wire must be to create an adequate number of windings for the desired induction effect. If we use too small a diameter, the coil becomes quite tall. With too

Hobby electronic stores sell small spools of magnet wire at the lengths required for this project. The tube I used for the secondary coil for the prototype was sold as a container for multiple spools of thread, but any similar size tube (such as PVC

FIGURE 2.

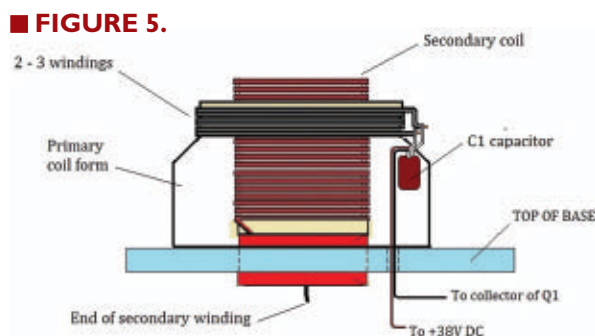
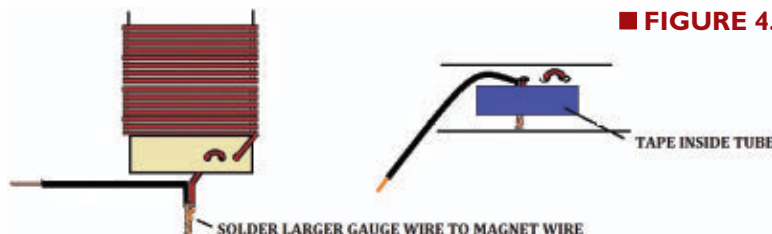
Begin construction by taping one end of the wire to the form approximately one half to one inch from the end, leaving a six inch length of wire. Wind the turns with tension applied such that the windings are not loose and spaced as close together as possible as shown in **Figure 3**.

Cut small segments of tape ahead of time to use for any breaks in the winding process. When finished winding the secondary coil, tape the end winding in the same way as the beginning winding. Remove any temporary tape used in between the first and last winding, and spray the coil with an aerosol clear acrylic and let it dry. (Most acrylic sprays require about 30 minutes of dry time.) Drill three small holes at either end of the tube to create a strain relief that will allow connection of stranded wire to the magnet wire as shown in **Figure 4**.

The primary coil is very easy to construct. Surface area is the most important metric which can be accomplished using either large diameter insulated wire or copper tubing. The position of the primary can be at any point around the secondary.

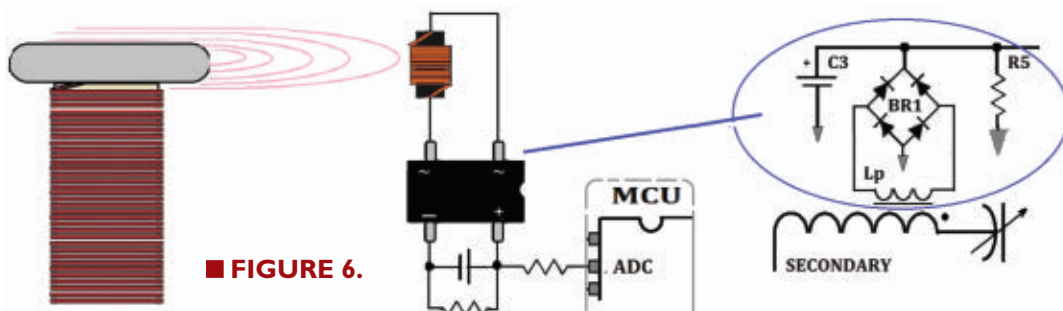
Looking at the **schematic**, you can see that the primary and secondary coils are polarized. If you wind the secondary in a clockwise manner, the primary coil will need to be wound clockwise also for induction to occur. Look in the PVC section of your local hardware for primary coil forms. A good choice for the form would be an expansion or reducing coupling of about two to three inches. Wrap two to three turns of wire used for 120V power cord around the form as shown in **Figure 5**.

The coil pair is made to resonate by the oscillating DC potential applied to the primary coil. This is accomplished through the collector connection made to one end of the primary to the circuit input power supply. Power is created by regulating the output of a 24 volt transformer, and is used to power the primary coil. The base of the driving transistor is switched on and off by pulse width modulation (PWM) using the C2 pin on the PIC. A simple program running on the PIC sweeps

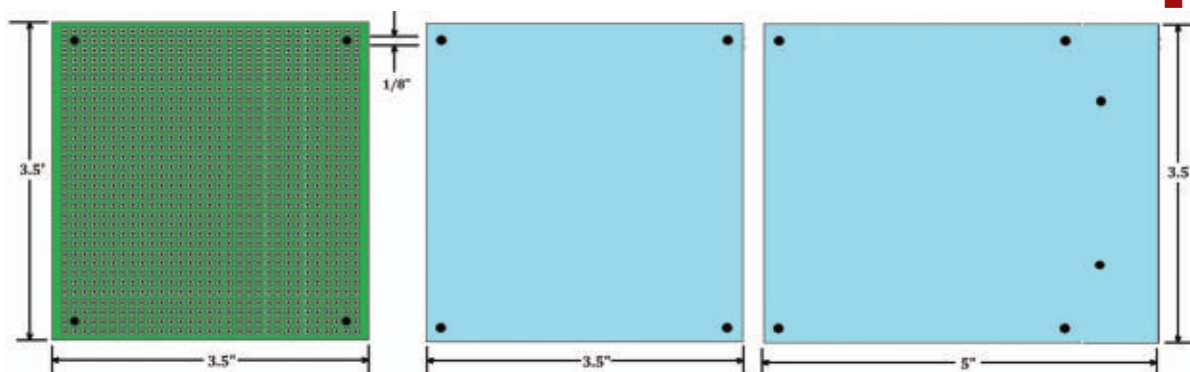


a frequency range as it samples the output of a small ferrite core inductor mounted under the secondary.

Current is induced in the small pickup coil which is located within the EM field of the secondary, acting as a power receiver as shown in **Figure 6**. Feedback from the voltage that develops on the small inductor is fed to an ADC (analog-to-digital controller) pin on the controller. The supply voltage acts as a reference to the incoming value and divides this analog voltage into a digital range from zero to 255. If the supply is three volts, an incoming voltage of 1.5 volts reads 127, or half the reference voltage. As soon as the regulated output from the receiver



■ FIGURE 7.

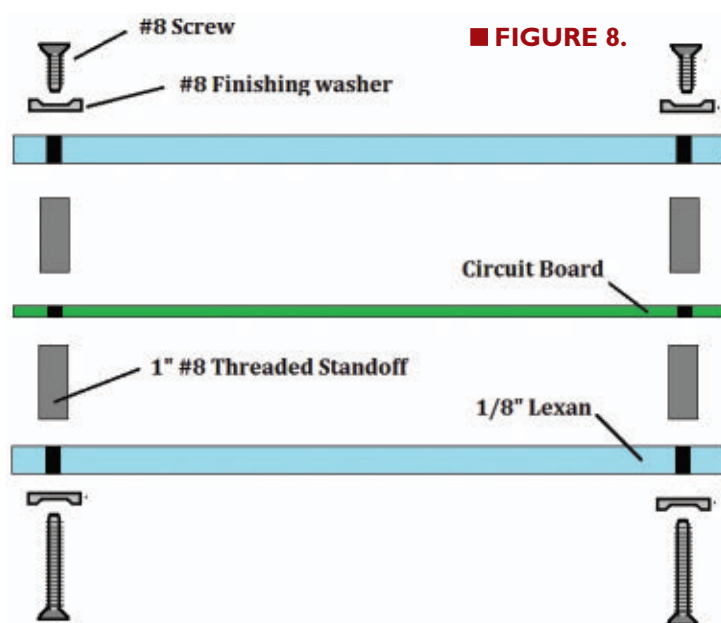


reaches the same or greater value of the potential applied to run the microcontroller, it locks on to that frequency. If you construct the coils close to the specifications given, the primary will induce a great deal of power with a considerable bandwidth of about 3 kHz or greater on

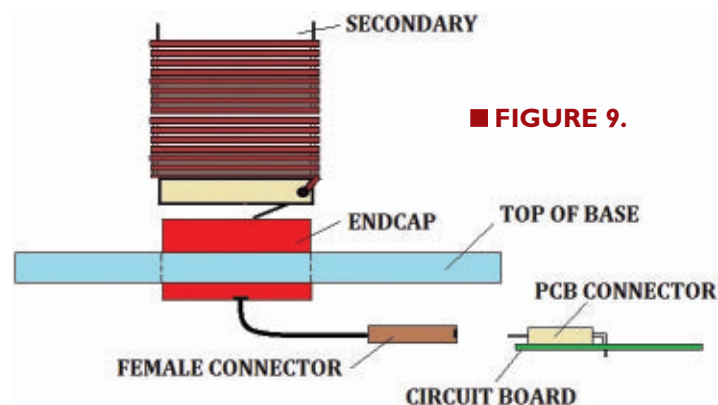
either side of the resonant frequency. You should notice wireless power to the CFL starting as low as 700 kHz and as high as 3 MHz.

Circuit Construction

■ FIGURE 8.



■ FIGURE 9.

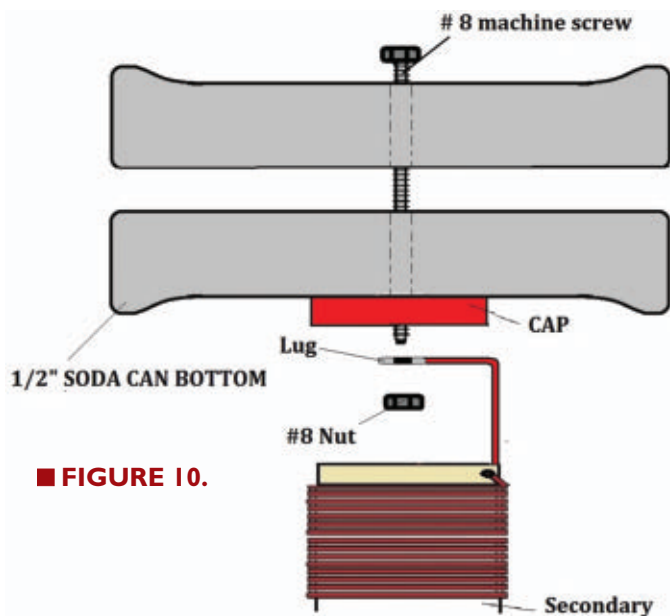


The entire circuit can be made to easily fit on a raised platform that supports the primary and secondary coils. To construct a design like the one shown here, you will need the following pieces of hardware and tools. Start the design by cutting a 3.5 inch square piece of prototyping board and drill 1/8 inch holes in the corners as shown in **Figure 7**.

The base for the design is made from 1/8 inch Lexan or similar plastic material. Cut two pieces of the Lexan into one 3.5 inch square and one 3.5 inch x 5 inch size, and drill 1/8 inch holes in the corners by using the printed circuit board (PCB) as a guide. Using eight one inch threaded aluminum standoffs, mock assemble the PCB and Lexan squares as shown in **Figure 8** to make sure everything aligns properly.

The one most important consideration is the inclusion of the power supply. I designed the prototype for this design with an onboard transformer which requires tall enough standoffs that provide clearance. I also made provisions for an off-board supply by using a rectified input jack as noted in the **schematic**. This allows you to experiment with different supplies to the circuit. The tube for the secondary coil is mounted to the upper Lexan piece by gluing one of the end caps to the 1.5 inch hole cut in the plastic as shown in **Figure 9**.

Drill a small hole in the bottom of the tube cap and feed the six inch length of wire through it for connection to the PCB. I used a SIP male and female connector pair for convenient connection to the circuit board. I constructed the toroid for the coil from the bottom of two soda cans. To create the toroid, saw one inch of the bottom of two aluminum cans and sand off the labels from both halves along with the plastic coating from the insides. Drill a hole



■ FIGURE 10.

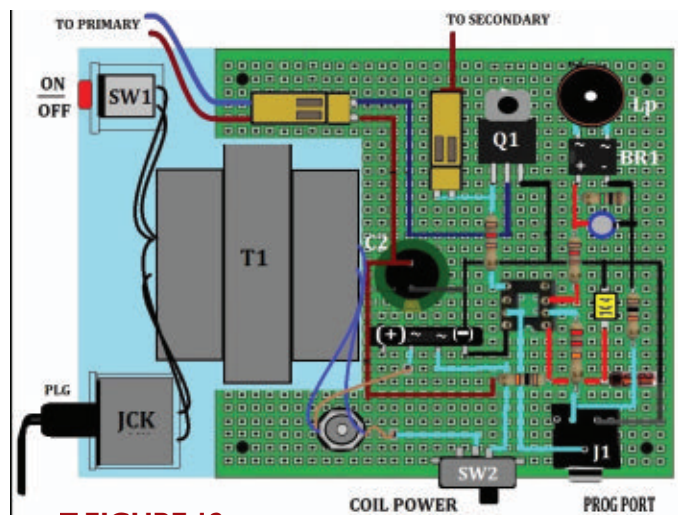
in the center of both bottoms; when the cans are fitted together, you should be able to test connectivity from top to bottom using a multimeter. Cut a corresponding hole in the center of the other tube end cap and assemble the two bottom halves of the cans as shown in **Figure 10**.

A solder lug is a convenient way to make the connection to the end of the coil. After connecting the toroid, check for connectivity from the bottom wire of the secondary to the top of the toroid. The resistance should be the same as the ohm value of just the coil by itself. You can attach a wire or metal piece with sharp edges to the top-most point on the toroid to provide a breakout point for corona discharge if you like (illustrated in **Figure 11**). If you construct the primary to secondary geometry correctly, the corona should self-discharge without a nearby ground path.

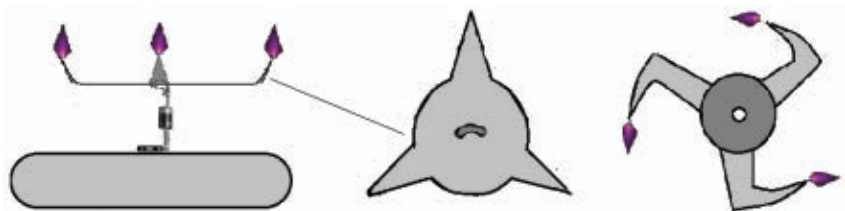
Try attaching thin pieces of aluminum or tin geometries that have intentionally designed sharp points for the best corona effects. Use wire or a stiff lead component to elevate the metal shape. The intensity of the corona is a function of the capacitance of your toroid. A pinwheel design using a circular array of points will actually spin as charge leaves the sharp edges.

Constructing the Printed Circuit Board

The parts layout for the PCB is not critical. If you mount the 25V transformer on the base, you may want to contour the PCB for it to fit as shown in **Figure 12**. The power jack located in the lower left corner of the PCB is a parallel



■ FIGURE 12.



■ FIGURE 11.

connection to the 25V AC output of the transformer. It can be used for an alternate input voltage source.

The 120V input to the transformer connects to the circuit using a molded two-pin connector. The RF may interfere with programming, so you may want to open the +V supply to the coil using an optional SPST switch shown in the **schematic**. This allows a programming voltage to be present on the controller, but disables the coil output. The C4 capacitor that connects across the coil should be placed as close as possible to the primary. This large capacitor can be integrated into the coil form used for the primary. This high voltage capacitor is important and for best operation, it should be able to withstand a minimum of 1,600V potential.

The capacitor is constructed from metalized polypropylene and is specially designed for horizontal resonance circuits for color TVs and monitors. These capacitors can be hard to find and expensive to buy, and are best salvaged from the circuit boards of a monitor or TV.

How to Demonstrate It

A high voltaic potential will accumulate on the capacitive structure on the end of the secondary coil. By convention, it is usually constructed in the shape of a

```

setfreq m32      'remark setfrequency 32 Mhz
b2 = 3 : b4 = 8  'remark set initial values for b2 and b4 (2 Mhz, 50% duty)
symbol FRQ = b2: symbol DUTY = b4 'remark symbolize variables FRQ and DUTY
Do while b0<255  'remark loop until voltage on pickup coil is maximum
readadc c.4,b0  'remark read the voltage on pickup coil
Gosub TESLA     'remark jump to PWMOUT routine
Loop
TESLA:
DUTY = DUTY + 1 'remark increment DUTY cycle
FRQ = FRQ + 1   'remark increment FREQUENCY
PauseUs 1200
PWMout 2, FRQ, DUTY 'remark frequency generated on pin C.2
PauseUs 1200
If b0>150 then TESLA 'remark If voltage on pickup-coil is MAX, repeat loop
return

```

■ FIGURE 13.

Software

The code driving the coil is the small Basic program in Figure 13.

Voltage Check

1) The first voltage point indicated as TP1 on the **schematic** should be

approximately 38V DC. If no voltage is present, check that there is 120V AC on the primary side of the transformer. If the 120V AC is present, check for continuity to the AC terminals of the bridge rectifier marked ~. There should be an AC voltage of approximately 24V AC. If this voltage is present, check for continuity to the electrolytic C2. If the capacitor is connected correctly, the rectifier may be defective.

2) The second voltage point labeled TP2 should measure about 3.7V DC that powers the 08M2. If no voltage is present, check the orientation of the zener diode Z1. If the diode is installed correctly, check that resistor R3 is connected to the +V DC potential on TP1.

Conclusion

Even though the design has been intentionally engineered to be safe, there is always the potential for electric shock. If you are unsure about working with any portion of the circuit, please seek help with the design from a more experienced person. I hope this project will spark your interest. **NV**

sphere or toroid to avoid sharp edges. Most all Tesla coils demonstrate beautiful electrical arcs or coronas that are discharged from the toroid by using some kind of breakout point. The size of the toroid you construct will make a difference in how the electricity is discharged. If you use a smaller toroid, electricity will be discharged more rapidly, but the arcs will not be as long. If you use a larger toroid, electricity will be discharged less rapidly, but the arcs will be much longer.

The next most popular demonstration would be the wireless lighting of gas filled tubes such as fluorescent or even neon. I have purposely kept the power low on this design for safety, but you should be able to see one to three inch coronas from the toroid if you attach a breakout point. With as little as 12V input to the primary, a five foot CFL glows very brightly, drawing as little as only 100 mA. At 30 volts, the coronas are very pronounced and make a hissing sound, and just begin to influence nearby electronics. There is little risk of an electric shock but there may be a risk of RF burns from nearby metals. The material is not heating but is actually arcing to the skin at high frequencies.

"Put the Spark back into your electronics"

Solid State Tesla Coils **Plasma Speakers** **LED Candle Kits** **High Power LED Kits** **Eastern VOLTAGE RESEARCH**

Create 4-5 Foot Arcs and music with the Plasmasonic® Musical Tesla Coil
MIDI Compatible

Commercial Grade

Designed and Built in the USA

Human Conductivity Kit Body Light 1.0

RGB Mood Light Kits

LED Sequencer Kits 24 MODES!

Advanced Strobe Kit (2) 3 Watt CREE LEDs

SSTC 1.0 Kit Solid State Tesla Coil \$99.99

EXTREME Joule Thief 3 Watt CREE LED

High Power LED Flame CR2450 Battery

USB Battery Charger For CR2450 Batteries

QR Code

EasternVoltageResearch.com

PicoScope[®] 2200A Series

Like a benchtop oscilloscope, only smaller and better



- Up to 200 MHz bandwidth
- 1 GS/s sampling • Advanced digital triggers
- AWG • Serial decoding • USB powered
- Ultra compact design

PicoScope[®] 5000 Series

Flexible resolution oscilloscopes

- Resolution from 8 to 16 bits
- 200 MHz analog bandwidth
- 1 GS real-time sampling
- 512 MS buffer memory
- 200 MS/s AWG



PicoScope	PicoScope 5442	PicoScope 5443	PicoScope 5444
Channels	4	4	4
Bandwidth	All modes: 60 MHz	8 to 15-bit modes: 100 MHz 16-bit mode: 60 MHz	8 to 15-bit modes: 200 MHz 16-bit mode: 60 MHz
Sampling rate - real time	1 GS/s (8-bit mode)		
Buffer memory (8-bit) *	32 MS	128 MS	512 MS
Buffer memory (≥ 12-bit)*	16 MS	64 MS	256 MS
Resolution (enhanced)**	8 bits, 12 bits, 14 bits, 15 bits, 16 bits Hardware resolution + 4 bits		
Signal Generator	Function generator or AWG		

2 Channel models also available * Shared between active channels ** Maximum resolution is limited on the lowest voltage ranges: ±10 mV = 8 bits • ±20 mV = 12 bits. All other ranges can use full resolution.

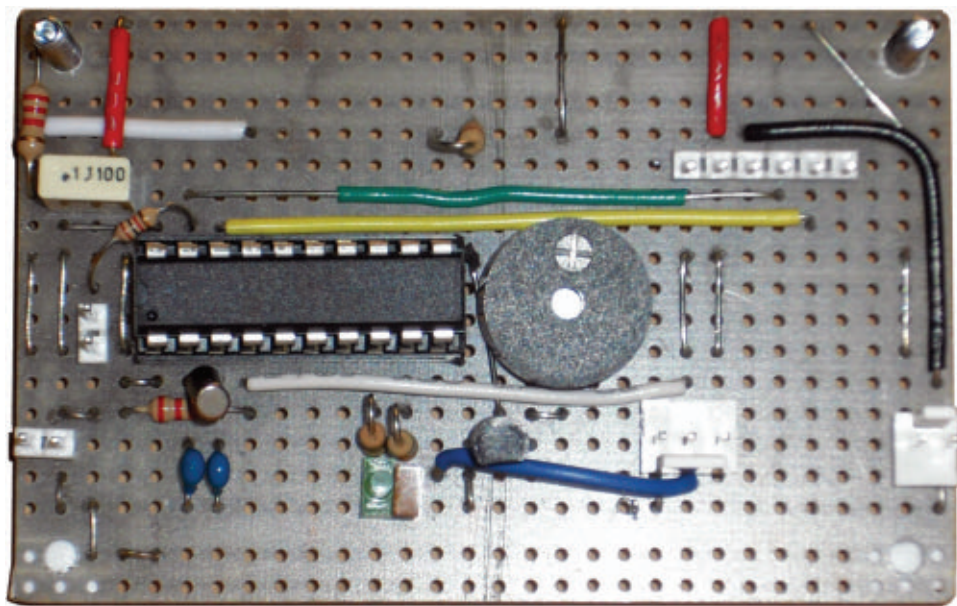
ALL MODELS INCLUDE PROBES, FULL SOFTWARE AND 5 YEAR WARRANTY. SOFTWARE INCLUDES MEASUREMENTS, SPECTRUM ANALYZER, SDK, ADVANCED TRIGGERS, COLOR PERSISTENCE, SERIAL DECODING (CAN, LIN, RS232, I²C, I²S, FLEXRAY, SPI), MASKS, MATH CHANNELS, ALL AS STANDARD, WITH FREE UPDATES.

www.picotech.com/pco518

DRYER MINDER

By Jim Lacenski

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/april2014_Lacenski.



■ FIGURE 1.

Shortly after our new clothes dryer arrived, I appreciated how quiet it was compared to the old one. I also noticed that the lint trap on the dryer plugged up rapidly; the lint trap surface area was only about a third of the previous dryer. While the dryer has a moisture sensor, the lint trap consistently plugs up when drying a load of towels, and the moisture sensor rarely activates. I knew it was time for a Dryer Minder — a simple alarm that can emit a beep 20 minutes into the drying cycle as a reminder to check the lint trap and clean it out if necessary. This enables a shorter time to dry clothes, less energy cost, and longer dryer life.

For design goals, I wanted to: 1) have no electrical connection to the dryer; 2) use an inexpensive PIC microcontroller; and 3) minimize wiring and build time, including debug time. To meet goal 1, I decided that a current sense transformer would meet the “no contact” criteria. For goal 2, I decided to leverage the PIC16F687 due to its real time clock-able timer, sufficient pins to keep the programming and project pins mostly separate, and low cost. Goal 3 is met by use of a solder breadboard, the useful serial output of the PIC16F687 for debugging, and other test features. **Figure 1** shows the main Dryer Minder printed circuit board (PCB), ready to be mounted externally from the dryer.

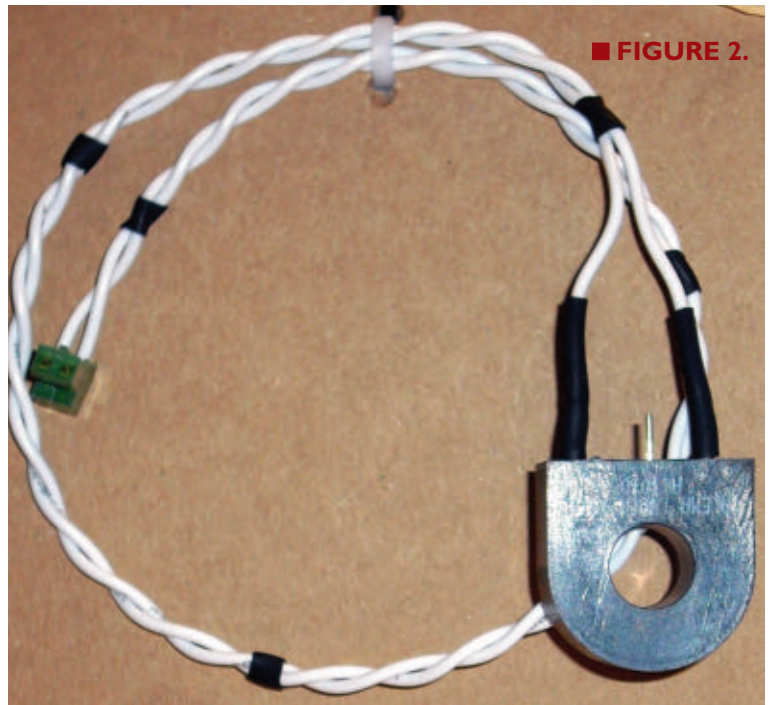
CAUTION: While the Minder has no electrical connection to the dryer, it does require opening the control panel which has hazardous voltages. **Be sure that the dryer is unplugged** when installing the portions of the project which reside in the dryer. For my recent model GE dryer, removing the back panel screws, tilting the control panel forward, and sliding the panel to the side was sufficient. YouTube has many instructional videos on how to open common dryers, so finding out how to open the panel was trivial.

The “Aha!” moment of the project was realizing that I did not need to sense the full dryer current, but instead could sense motor current and use that to enable the Minder. An economical current sense transformer slipped over a motor power wire makes sensing the dryer run condition easy.

The next task was to interface the current transformer to the PIC. At typical motor current, the current transformer provides about two volts of signal. A simple voltage doubler circuit boosts this voltage, turning on a transistor which acts as a switch. This is wired on a “sense board” which is mounted on the dryer back panel in the control area.

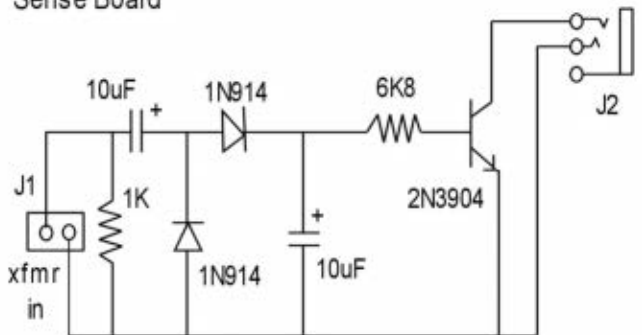
The main Dryer Minder board contains the control PIC. To reduce power during standby, timer 1 of the PIC is configured to run as a clock using an interrupt, where the PIC is asleep most of the time and only briefly wakes up each second to check for input. This had more reliable results than using the interrupt on change abilities of the PIC. While this does incur some standby power, this allows the project to expand to other sensors. For example, a future version may also sense from a humidity sensor in the laundry room which will be helpful when drying clothes with fan-assisted air drying.

Making the system easy to test is made possible by four features: 1) the inclusion of a Heartbeat LED that is on when the processor is not in sleep mode; 2) a Run LED which is on when the Run condition is sensed; 3) a test mode that will turn the beeper on in 20 seconds instead of the usual 20 minutes; and 4) output of the Run count



■ **FIGURE 2.**

Sense Board



■ **FIGURE 3.**

(in seconds) to the PIC UART serial output for viewing on a serial LCD display.

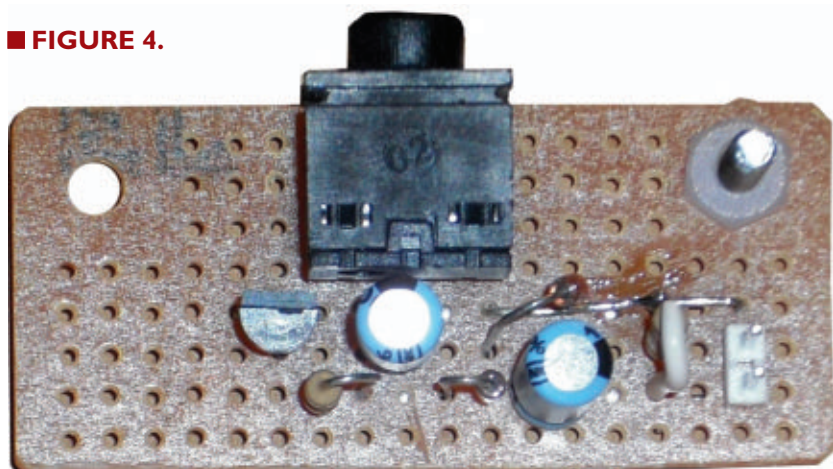
Power for the Dryer Minder is from an inexpensive five volt cell phone charger I found at a local Goodwill store.

Construction

Construction starts with the current sense transformer. Two wires approximately 12” in length are connected to the outermost pins of the transformer, tightly twisted together for noise immunity, and press-fit into a two-pin connector (TE Connectivity); refer to **Figure 2**.

Next, wire the sense board. This is the circuit that takes the voltage from the current sense transformer and produces the switch output. The circuit diagram is shown in **Figure 3**.

■ FIGURE 4.



An inexpensive perf board was used for the prototype. This is cut to size to fit in the limited space of the control panel. The sense board connects to the current transformer with the two pins on the lower far left. Output to the main board is through the 3.5 mm jack. The sense board is shown in **Figure 4**.

The Dryer Minder main board is next. The circuit diagram is shown in **Figure 5**. J1 is the sense input from the sense circuit. The oscillator crystal for the low power clock is seen connecting to pins 2 and 3. The Run and Heartbeat LEDs connect to pins 8 and 9. The beeper is switched through the PIC2N3904. J2 pins are used to activate test mode, and J3 is the serial output connector for test use. The code for the PIC is available at the article link. The

program was developed using the HI-TECH C compiler/Pro edition in free mode.

Note the few locations where traces need to be cut. These are identified by the thick black rectangles in the layout diagram, such as near the Run and Heartbeat LEDs, and the J3 (serial out/debugging) connector. The area to the lower right is intentionally left bare for future expansion. Refer to **Figure 6**.

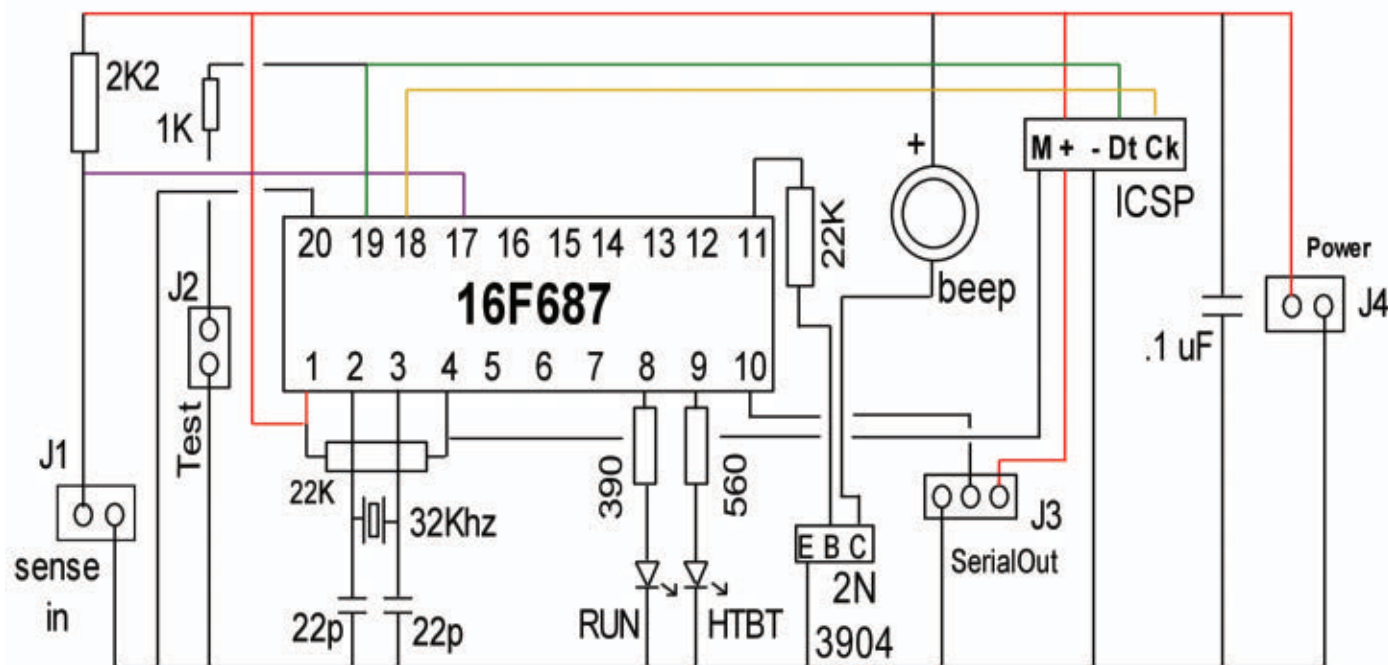
Programming and Testing

Programming is through the ICSP™ connector. I programmed the PIC using a PICKit 3. Once the program starts to run, the beeper will turn on for about two seconds before the PIC is put to sleep. The PIC will wake each second, flashing the Heartbeat LED. This is an indication of a successfully loaded and running program.

Jumper J1: This condition simulates an active low from the sense board, as would occur during “dryer on.” The Run LED will light every time the Heartbeat LED is on when J1 is jumpered/set low. If you have connected the serial LCD to the serial output (J3), the count in seconds will appear on the display while in Run mode.

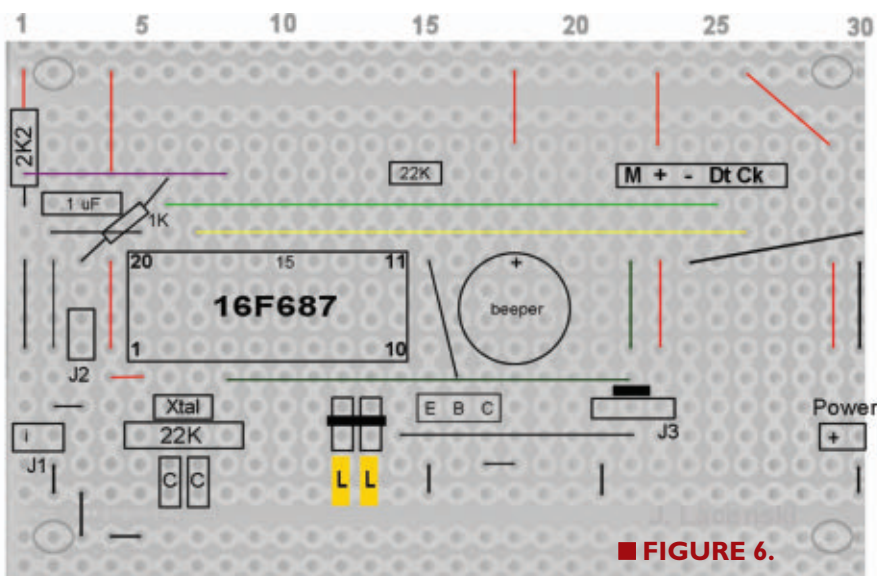
When leaving the connection to the PICKit 3, the PICKit 3 causes RA0 to be pulled low, activating Test mode. The beeper will sound 16 seconds into the test cycle if the PICKit 3 is still connected and there’s a jumper

■ FIGURE 5.



Installation

Installation of the current sense transformer was as simple as unplugging the brown wire's spade terminal from

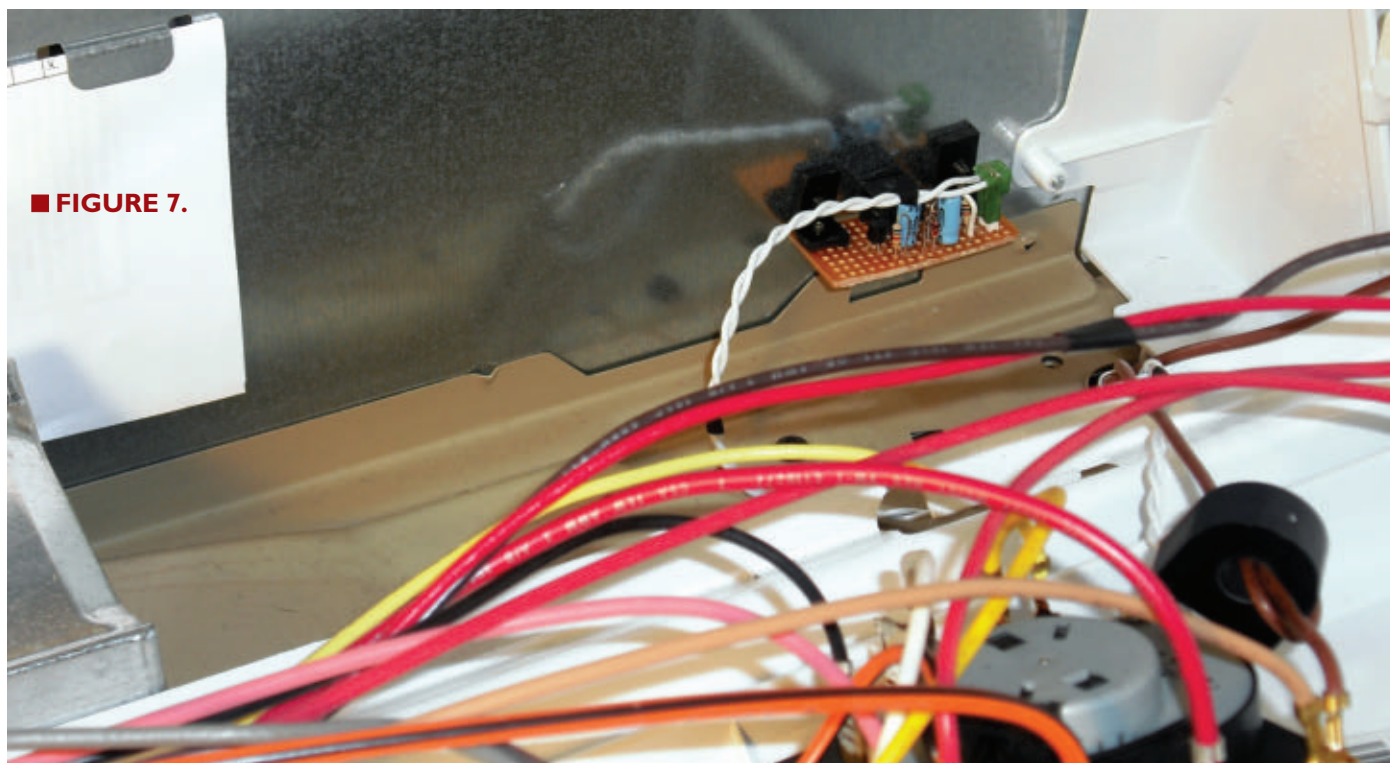


■ **FIGURE 6.**

the dryer timer and slipping the wire through the opening in the current sense transformer. The spade terminal was then reattached to the timer. All simply done, without cutting the wire. These instructions are for the GE dryer that I installed the circuit into. Check your dryer's wiring diagram for the wire that supplies current to its motor.

ITEM	PARTS LIST	PART #	SOURCE
Current Sense Transformer: 10 amp current transformer		1295-1102-ND	Digi-Key
Sense Board: Two-pin connector 1K, 6.8K resistors 10 μ F 16 volt capacitors 1N914 or equivalent diodes 2N3904 transistor 3.5 mm jack Perf board, single hole pad		A1921-ND CP1-3524NG-ND PC-2	Digi-Key Digi-Key, All Electronics Digi-Key All Electronics
Main Board: Two-pin connector (two used) Three-pin connector (serial out) Two-pin connector (power) Two-pin female socket connector (two) 2.2K, 1K, 22K (2), 390 ohm, 560 ohm resistors PIC16F687 microcontroller 12.5 pF capacitors (two) LEDs, thin profile 8 x 2.5 mm Crystal 32.768 kHz 12.5PF 2N3904 transistor Beeper 3-18 volt, small Six-pin connector for ICSP .1 μ F 100 volt capacitor Solder breadboard 400 point Recommended: Tool for TE Connectivity MTA-100 connectors		A1921-ND CON-243 CON-242 A30827-ND All Electronics PIC16F687-I/P-ND LED-171 X1123-ND SBZ-204 A31116-ND MMC-104 SB-400 A9982-ND	Digi-Key All Electronics All Electronics Digi-Key Digi-Key Digi-Key All Electronics Digi-Key All Electronics Digi-Key All Electronics All Electronics Digi-Key
Miscellaneous: Two conductor plus shield cable		CB-223	All Electronics

■ FIGURE 7.



The 3.5 mm jack which I used for the project requires that the sense board be mounted with small right angle brackets and holes drilled in the dryer back panel for the jack, as well as the mounting screws. Plastic brackets were fabricated and holes drilled to enable mounting. Other jacks with threaded barrels can simplify mounting. **Figure 7** shows the sense board mounted to the back panel, with the current sense transformer in the foreground.

A two-conductor plus shield cable connects the sense board to the main board. There is no electrical connection to the shield at the dryer to avoid potential ground loops. Be mindful to connect the cable wires so that the collector of the transistor on the sense board connects to the signal input of the main board. The shield should connect to the common of the main board.

The Dryer Minder board is mounted next, external to the dryer. I use a shaped plastic plate with clearance for all parts and connectors of the main board. I recommend mounting the main board so that the Heartbeat and Run LEDs are easily seen. Connect the five volt power supply to the main board.

Final Testing

With the main board powered up, the Heartbeat LED should be seen to briefly flash each second. Start the dryer; the Run LED should flash in tandem to the Heartbeat. At 20 minutes less four seconds, the Minder should beep, alerting you to check the lint trap.

Other Applications

This project lends itself to many other applications besides green technology. Since the easy to edit C code includes serial output via the PIC USART Tx pin, display or telemetry applications are plentiful. For example, a running total of on time for a machine or appliance could be easily implemented using a serial LCD for display, such as the SparkFun LCD-09395 which was used during development. By decreasing the USART data rate, the SparkFun WRL-10534 and WRL-10532 can be used for remote telemetry. I use these for wireless control of a fan system.

An alternate telemetry application would be to sense open and close times of a remote gate. The PIC has rich analog input, such as for sensing current of water heater elements or temperature with an analog temperature sensor. Port C — used for the Heartbeat and Run LEDs — has two additional inputs/outputs that can be used for alarms, moisture sensors, and other peripherals.

Whatever the application, interference with the real time clock (one second counter) connected to pins 2 and 3 or the ICSP header connections should be avoided. Be careful to not push too much code into either the interrupt handler (used by the clock) or the main loop. I found that longer messages to the serial display could have timing issues at the default (9600) baud rate of the serial display when using un-optimized code. If you need more component room for expansion, All Electronics sells a larger solderable breadboard suitable for more advanced projects.

What do you have in mind for this circuit? **NV**

Beginner's Guide Complete Combo!



Combo Price \$229.95

For complete details, visit our webstore @ www.nutsvolts.com.

Call to
order at
**1-800-
783-4624**
or go to
**www.
nutsvolts.
com**

Feedback Motion Control

The Old Way

- 1) Build robot
- 2) Guess PID coefficients
- 3) Test
 - 3a) Express disappointment
 - 3b) Search Internet, modify PID values
 - 3c) Read book, modify PID coefficients again
 - 3d) Decide performance is good enough
 - 3e) Realize it isn't
 - 3f) See if anyone just sells a giant servo
 - 3g) Express disappointment
 - 3h) Re-guess PID coefficients
 - 3i) Switch processor
 - 3j) Dust off old Differential Equations book
 - 3k) Remember why the book was so dusty
 - 3l) Calculate new, wildly different PID coefficients
 - 3m) Invent new, wildly different swear words
 - 3n) Research fuzzy logic
 - 3o) Now it is certainly not working in uncertain ways
 - 3p) Pull hair
 - 3q) Switch controller
 - 3r) Re-guess PID coefficients
 - 3s) Switch programming language
 - 3t) Start a new project that doesn't need feedback control
 - 3u) See parts in box. Feel guilty. Go back to old project
 - 3v) Start testing every possible combination of PID coefficients
 - 3w) Apply eye drops to red, blurry, sleep-deprived eyes
 - 3x) Wait, it's working!
 - 3y) Decide not to do any more projects that require control systems
 - 3z) Wonder why someone doesn't just make a thing that tunes itself

The Kangaroo x2 Way

- 1) Build robot
- 2) Press Autotune
- 3) Get a snack

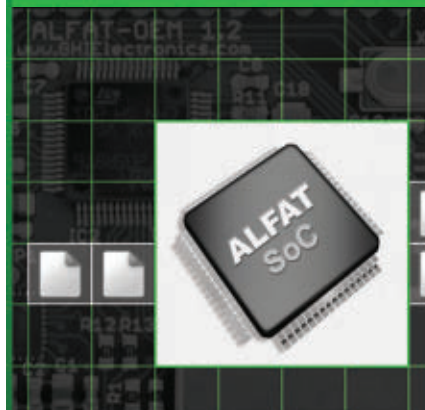
**Kangaroo x2
adds self-tuning
feedback to SyRen
and Sabertooth motor
drivers.**

\$24.99

www.dimensionengineering.com/kangaroo



File System Solutions



With **ALFAT SoC** You Can

Use FAT/FAT32 with
Long File Name

Use UART, SPI or I2C serial
interface to access files.

Use SD-Reader Mode
allowing the ALFAT SoC
to act as an SD Card Reader.

Access files on two USB
drives and an SD card
simultaneously

Use high speed USB and
4-bit SD card interfaces.



The ALFAT SoC is also available
as an OEM board.

GHIElectronics.com

25 YEARS Beta LAYOUT create:electronics

Create your own Solder Reflow station

• Reflow Controller:	\$ 315.00
• Standard Oven:	\$ 60.00
• PCB:	\$ 30.24*
• Laser Stencil:	\$ 0.00
Total Cost	\$405.24

* Based on 5 PCBs at 4" x 4";
2 Layer with Mask & Silk, ordered at
www.pcb-pool.com

Laser Stencil **FREE** with PCB-POOL® order!

Order online at
www.beta-eSTORE.com

Just launched in the USA

eSTORE
Beta LAYOUT



THE DECADE BOX REVISITED

By Frank Muratore

Post comments on this article and find any associated files
and/or downloads at [www.nutsvolts.com/index.php?](http://www.nutsvolts.com/index.php?/magazine/article/april2014_Muratore)
[/magazine/article/april2014_Muratore](http://www.nutsvolts.com/index.php?/magazine/article/april2014_Muratore).



As a *Nuts & Volts* reader, you probably design and build circuits (from time to time). So, you know that circuit simulation software can sometimes fail at getting an exact value for a component. Therefore, you might resort to trial-and-error substitutions. This can be an arduous task at best.

A far better approach might be to have a device which is adjustable over a wide range that will provide the necessary value. When looking for a resistance, you might be tempted to use a potentiometer; once the circuit functions, read the value and substitute with a fixed component. So, you obtain a 10-turn pot to facilitate fine adjustment.

This approach works as long as the power requirement of the pot is not exceeded. Most 10-turn pots will only take limited power. So, you're looking at having to spend a lot of money for a suitable wattage unit – especially if you're designing power supply circuits (where the power requirements of the components can be quite large).

A far better compromise is to have a "box" with suitable pots that can take a fair amount of current, yet be adjustable over a wide range. That's where a decade box can save you.

The original decade boxes consisted of a number of fixed precision resistors and switches that would add or subtract the selected values. These boxes were not only costly, but tended to be very large.

The other day, I had an idea to duplicate the functionality of the boxes (without the size or price). The advent of cheaper and cheaper digital meters and the accuracy they bring lends itself to a much more efficient alternative.

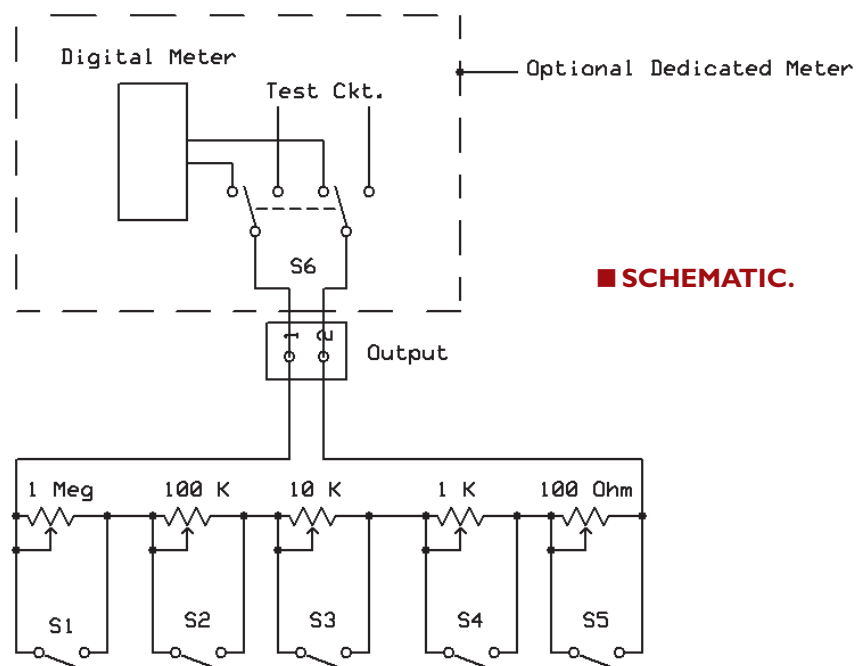
Previous decade boxes had faceplates with the resistance values printed on them so you could adjust the box and then add the values to obtain the resistance needed. My approach here is one of simplicity. Instead of having the printed values on the panel and adding up the values, why not adjust the box and then measure the total using a digital meter!

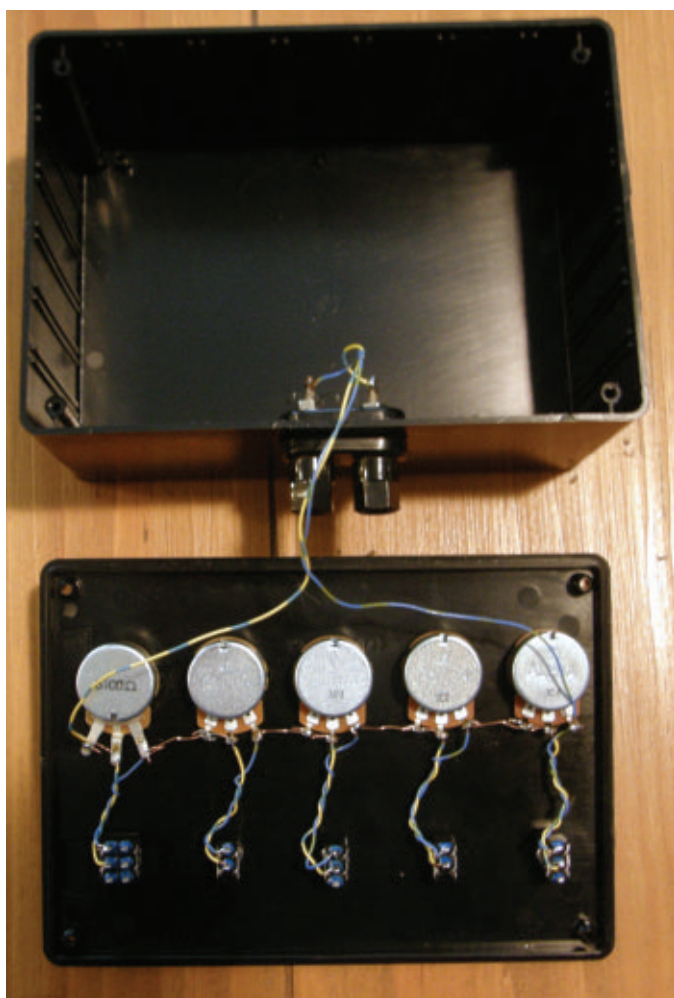
So, I set out to build my box. For simplicity and also low cost, I used linear taper potentiometers. These devices take up little space and can be had for cheap. In order to add or subtract the resistance values, I use common single-pole switches. The box can be as simple or complicated as you wish. I have limited the total value of resistance to be 1.1111 megohms (this is derived by adding one meg, 100K, 10K, 1K, and 100 ohms together). However, if you like, you can add resistance to either end to tailor the box to your specific needs.

As stated above, the measurement device is a simple digital voltmeter with resistance scales. The five-way binding post connection allows the meter to be connected when needed, so as not to tie it up. These meters are so cheap today, that you might want to incorporate a

Table 1. Standard Resistor Values (15%)

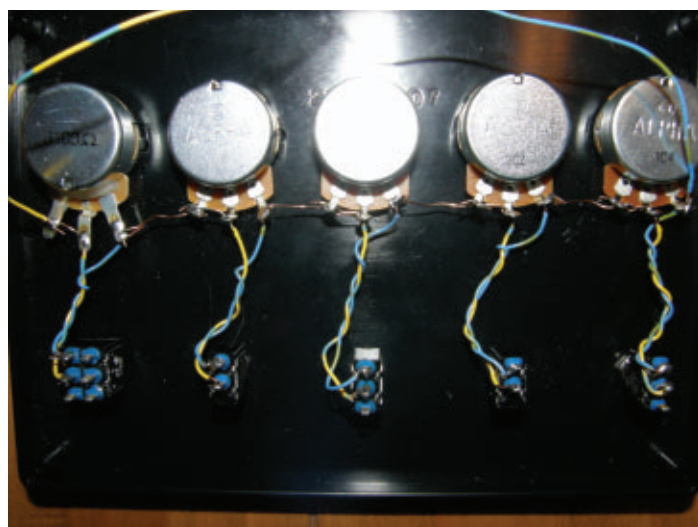
1.0	10	100	1.0K	10K	100K	1.0M
1.1	11	110	1.1K	11K	110K	1.1M
1.2	12	120	1.2K	12K	120K	1.2M
1.3	13	130	1.3K	13K	130K	1.3M
1.5	15	150	1.5K	15K	150K	1.5M
1.6	16	160	1.6K	16K	160K	1.6M
1.8	18	180	1.8K	18K	180K	1.8M
2.0	20	200	2.0K	20K	200K	2.0M
2.2	22	220	2.2K	22K	220K	2.2M
2.4	24	240	2.4K	24K	240K	2.4M
2.7	27	270	2.7K	27K	270K	2.7M
3.0	30	300	3.0K	30K	300K	3.0M
3.3	33	330	3.3K	33K	330K	3.3M
3.6	36	360	3.6K	36K	360K	3.6M
3.9	39	390	3.9K	39K	390K	3.9M
4.3	43	430	4.3K	43K	430K	4.3M
4.7	47	470	4.7K	47K	470K	4.7M
5.1	51	510	5.1K	51K	510K	5.1M
5.6	56	560	5.6K	56K	560K	5.6M
6.2	62	620	6.2K	62K	620K	6.2M
6.8	68	680	6.8K	68K	680K	6.8M
7.5	75	750	7.5K	75K	750K	7.5M
8.2	82	820	8.2K	82K	820K	8.2M
9.1	91	910	9.1K	91K	910K	9.1M





■ Inside wiring.

dedicated meter to the project. If you do this, remember to include a switch to disconnect the meter (until you want to read it). This will eliminate voltage from ruining the meter (when it is set to the resistance scale) and also improve accuracy by not having the meter input resistance



■ Close-up of pot and switch wiring.

affect the box value. Both circuits are shown in this article.

Construction

Construction of the box is straightforward and wiring is not critical (see the **schematic**). As you can see, I used three types of switches to construct the project. That's because I couldn't secure all single-pole single-throw switches at the time I wanted to build the unit. So, be flexible, and either use what you have on hand or order them.

Theory of Operation

Each pot is wired to a corresponding switch. The switch either "shorts" the pot or allows its resistance to add with the other pots (which are connected in series). The resulting total resistance appears across the binding posts. The precision is achieved by the measurement

process using the digital meter. One should note that since the pots are connected as rheostats (i.e., one side tied to the wiper electrically), if they are adjusted for minimum resistance it should be zero. Therefore, the switches would not be needed. However, as pots age, the contact resistance can increase from zero ohms. The switches further decrease this resistance so it becomes negligible.

PARTS LIST

Description	Part Number	Supplier
1 meg Potentiometer	023-640	Parts Express
100K Pot	023-634	Parts Express
10K Pot	023-628	Parts Express
1K Pot	023-624	Parts Express
100 ohm Pot	023-604	Parts Express
SPST Switch	275-634	RadioShack
*DPDT Switch	275-626	RadioShack
Dual Binding Posts	274-718	RadioShack
7x5x3 Case	270-1807	RadioShack
Five Control Knobs	274-415	RadioShack
Misc. Hook-up Wire	278-1222	RadioShack

*Optional For dedicated meter (refer to the schematic)

Parts-Express.com • RadioShack.com

Using the Box

Using clip leads, connect to the binding posts to place the box resistance across your circuit.

Put all switches in the “in” position. This will place all of the potentiometers in the circuit. Now, adjust the leftmost (one meg) pot until you get the desired result. If your circuit needs less resistance, put the one meg switch in the “out” position (eliminating the one meg pot). Now, adjust the 100K pot until the desired result is achieved. Again, if the resistance is still too much, put the next (10K pot) switch in the out position.

Keep repeating this procedure until you get the desired result. Once this is achieved, disconnect the clip leads from your project and place them across a digital meter (set to read resistance). Read the value, then look up that value in a chart of standard resistor values.

Table 1 shows an example chart.

Get the closest standard EIA (Electronic Industries Association) value and put this in your circuit. If a standard value isn’t close enough, use combinations of series or parallel resistors to get there. In some cases, a

potentiometer can be substituted for the value and adjusted to get the exact match.

So, build the box, and have an accurate variable resistance available to help your designs. **NV**



■ Box with meter attached.



**We Provide Complete Solutions
for Embedded Systems**

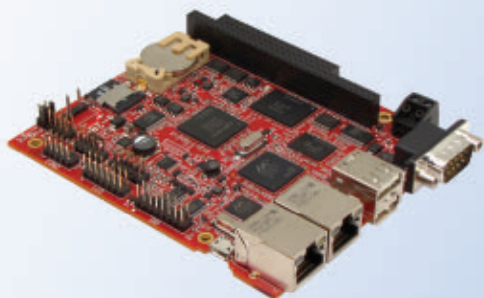
www.embeddedARM.com

(480) 837-5200

- 30 years in business
- Never discontinued a product
- Open Source Vision
- Engineers on Tech Support
- 45 Day Evaluation Period

Single Board Computers

TS-7250-V2



pricing
starts at
\$199
qty 1
\$165
qty 100

- Up to 1GHz ARM9
- Ethernet, USB, CAN, DIO, PC/104
- 512MB RAM
- Low power and sleep mode
- 4GB eMMC Flash
- Runs Debian Linux

Touch Panel Computers

Panel Mount or Fully Enclosed



Series starts at
\$409 qty 1
\$369 qty 100



- 5", 7" or 10" panels
- Fanless -20 to +70C
- Up to 1GHz ARM9 CPU
- Up to 512MB RAM
- Up to 4GB eMMC Flash
- 2x SD w/ DoubleStore
- 2x Ethernet, 2x USB
- CAN, SPI, I2C, DIO, WiFi
- RS485 2Wire-Modbus
- Fastboot Linux 2.6
- Interchangeable CPU cards

NEW! i.MX6 CPU coming soon

A FULL MOTION HOME SIMULATOR

By Walt Noon
admin@noonco.com

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/april2014_Noon.



Flying a small Cessna 172 on a short trip in Southern California, I experienced a radical stall after encountering unexpected wind shear conditions on approach to Riverside airport.

The little Cessna pitched hard to the left side literally throwing me up against the door.

As I struggled to back off power and neutralize controls, I realized I didn't have the altitude to recover from the impending spin.

Despite its normally docile nature, the 172 pitched violently nose down to the ground, and with my hand clenching the yoke and my knuckles turning white, we impacted the ground head-on at nearly 120 knots.

Fortunately for me, the entire crash happened in my garage in my own home-built, full motion flight simulator I called the "Virtual Flyer."

Of all the electronic projects I've been lucky enough to build over the years, I don't think anything has been as exciting as owning a machine that literally picks you up off the ground and immerses your senses *fully* in another world. Not to mention, a world in which you can fly!

The first version of my Flyer (discussed here) was built almost 15 years ago. Since that time, phenomenal new applications in software and hardware have emerged.

With wonderful programs like Google Earth, it is now possible to do what even the most advanced military computers could not do just a short time ago: allow you to fly in real time with real satellite photo images and weather anywhere in the world at a moment's notice!

In writing this article, I want to talk a little bit about the adventures of the Virtual Flyer, its creation, and motion simulator theory. I also want to discuss how you can get started building incredibly strong motion simulations just like I did.

Note: This article is not meant to be a precise step by step, bolt by bolt description of one simulator (which would be impractical short of a book length effort), but will give you a highly detailed overview of all the basic systems you will need to create your own flying simulator easily. In addition, I'm making all software open source and downloadable, and will provide videos of the machine to simplify your construction, as well.

Working on a full motion simulator will require some mechanical work, electronic work, and even a little programming, but surprisingly, it's not a great deal more difficult than many other *Nut & Volts* projects. I'm confident that the first time you step into your flying machine and leave reality for cyberspace, you'll agree it is worth the effort!

Don't forget, when your simulator is complete, you'll find me waiting for you on the Internet in a green cyberspace field somewhere, guns loaded and ready for combat — simulator to simulator.

Of course, you know where you'll be seeing me first ... in your rear window!

Virtual Reality Waits for No One

Fifteen years is an eternity in computer years.

Both the hardware and software described in this project have been greatly surpassed by newer computers and interfaces.

I'm certain you will have many ideas for how these can be updated, and I'll be including tips and tricks regarding simulation to help you do so.

That said, despite the older tech, the

simulator itself is still flying beautifully after all this time (having given over 40,000 rides at air shows and events). Should you choose to use the same hardware/software, you'll find yourself up and running in no time.

The Illusion of Flight/ Simulator Theory

In designing your own simulator, it's very important to understand how we perceive motion and our surroundings. Many aspects of making a good flying simulation are *surprisingly* counter-intuitive!

The first thing to take into account is how we as human beings sense motion. Our strongest sense of motion comes not from our inner ear, but from our visual senses.

If you were to stand in front of a movie screen staring straight ahead with no other visual cues and a flying scene was projected, most people will become so disoriented that without a handrail they will be unable to stand. However, if you were standing while watching the same scene on a TV, you would have no difficulty at all. This is because while the TV screen is projecting the same motion, your visual field isn't *filled* by that motion. Plus, you see other visual cues around it that aren't moving, so you are able to keep your balance.

For this reason, any simulator that does not *fully* enclose your vision will NEVER capture the feeling of flight ... not even to a small degree! While it may be fun to rock around in a moving chair, your brain will lock on to stable visual cues and all feeling of real flight will be lost.

So, the first rule of an immersive VR experience is: You must fully enclose your cabin or block all outside visual cues with virtual reality goggles (or the like).



A Quick Warning

Before we go into the nuts and bolts of this build, I wanted to offer a quick and gentle warning.

Even though a simulator is a ground-based machine, the forces at work are still quite significant. Pistons, electric actuators, and even hydraulics often exert thousands of pounds in force on single points.

In early tests of the larger simulator, a broken weld nearly caused an eight foot plunge into metal scaffolding on a test "flight" I made.

After scrambling free, I had to laugh since I assumed I could tell the story of being the only guy who nearly was killed in a *simulator* crash! However, whenever I've told this story to simulator enthusiasts, they've been quick to offer their stories along the same lines. It almost seems the rule rather than the exception.

So, be careful in your design and testing, and treat this like a real vehicle, which it is — even if most of the motion is in our minds!

The good news is we can also exploit this quirk of our own visual dominance to both simplify our simulator design and to radically increase the sensations in our ride. For example, most commercial simulator's range of motion is limited to only six degrees of motion. Yet, the riders inside *feel* as if they are moving as much as 360 degrees!

The reason for this is that the motion base only needs to provide a few moments of *acceleration* in any given direction. Then, the video the rider is watching takes over to make them feel as if the motion is continuing.

The effect is very striking, and I have had riders who were *absolutely certain* they had done a full loop in a simulator which actually only moved a few degrees.

Next, ideally, your simulator's cabin should be designed to move in conjunction with what is happening in your screen imagery. Motion should not be tied simply to joystick control as many simpler video games have done in the past. Locking motion to screen movements mimics how things feel in a real airplane, for example. Changes in thrust, rudder, and ailerons bring about unique attitudes in the airplane that are not dictated by stick position only.

Consider making a 360 degree turn in your car at 5 MPH or at 25 MPH. Though the wheel may be held in the same position, the forces you would feel would be entirely different at those two speeds. This is why simulator designers think in terms of accelerations and not movement or platform angles.

Another important reason to make certain that screen and platform motions are in sync is that most pilots will become nauseated within minutes when screen and platform are out of sync. It's possible in a closed-loop system to throw the screen and platform out of sync intentionally, however.

This always struck me as a possibly useful way to induce spatial disorientation and/or test anti-nausea protocols.

It's also a great way to make friends sick.

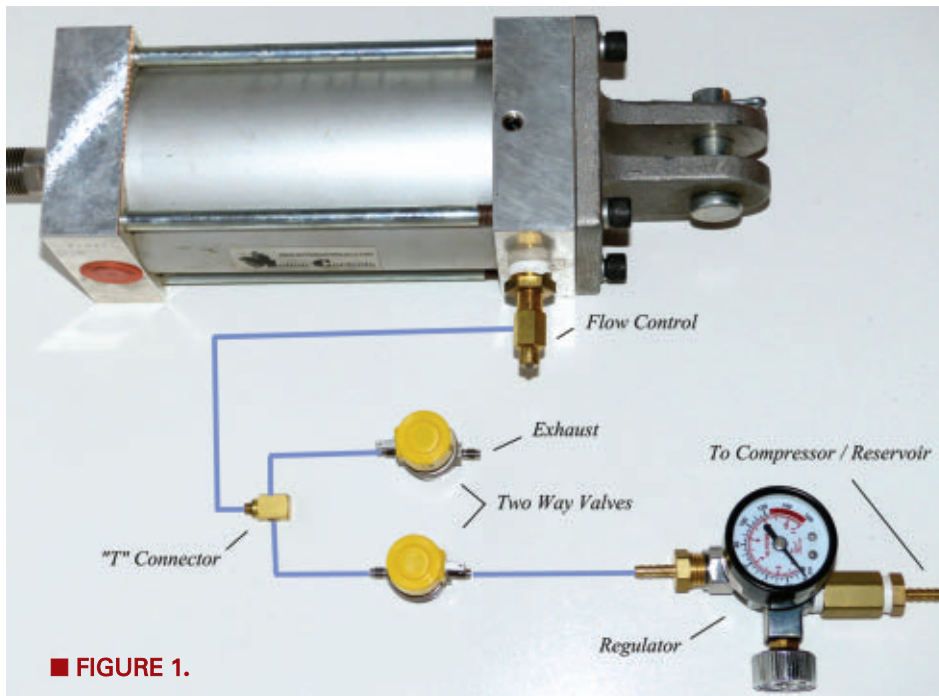
Deciding on an Actuator

There are three basic ways that most simulators are moved. These are with cylinders (hydraulic or pneumatic), servo/electronic, or manual weight shift operation. All of these have pros and cons.

For the purpose of this article, I'll stick with a pneumatic system as this is robust, relatively inexpensive, and something I've loved designing with for years. Since the electronics to be described represent a full closed-loop system, they should lend themselves to any actuator you choose to employ. So, feel free to use your favorite.

One advantage of pneumatics in flight simulation is that you are essentially riding on air shock absorbers. This means your flights will be glass smooth with no mechanical sounds or vibrations.

Each axis on which your simulator can move is referred to as



a DOF (degree of freedom). The simulator in this article requires two air cylinders, and is therefore a two DOF simulator (pitch axis and roll axis).

Once again, you are not limited to two DOF. Feel free to add rotational or even vertical acceleration actuators in your designs.

Figure 1 shows the simple hookup that will be used for each cylinder. Air from your compressor tank travels through a regulator to a 24V solenoid two-way valve. When this valve is opened, your cylinder will extend.

A second valve is connected to the system using a T fitting, and functions as an exhaust valve. When this valve is opened, the cylinder will retract. (The output from this valve can be connected to a muffler or long hose if you would like silent operation.)

Connected to the cylinder port is a needle valve for air flow control. This valve is important since it will determine the top speed with which your simulator will move, and prevent the air cylinder from extending too rapidly. (A wild ride is a good thing, but being thrown out of your seat is usually not ideal.)

Though I definitely recommend using all quality parts, I can't resist mentioning that one builder used ordinary sprinkler valves for air valves and home-made PVC cylinders to move his project, and it worked! So, it's even possible to build scrap box versions of this project.

Note: The simulator shown here used two cylinders for each axis. This was strictly because I happened to have smaller cylinders on hand when I built it. It is actually only necessary to use one cylinder for each axis by using a larger bore version.

In selecting your cylinder bore and stroke, determine the size you need based on your weight and your compressor's PSI rating. Cylinder force and PSI info will be available from the particular manufacturer.

Building the Cabin

Figures 2 and 3 show the first successful and easily built prototype which flew for many years. The design was intended to look like an "alien craft" for some air shows we did, but this cabin design looks just as great for a home simulator if simply painted black. The



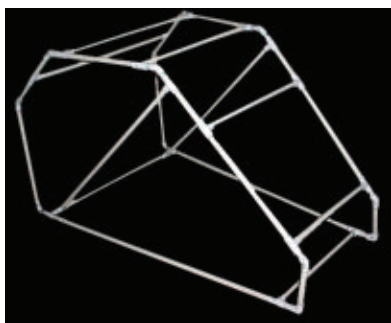
■ **FIGURE 2.**



■ **FIGURE 3.**



■ FIGURE 4.



■ FIGURE 5.

door is a simple curtain that was left off for the photograph to give a sense of size and rider position. **Figure 4** shows the inside layout with video screen, joystick, seat, and rudder pedals.

When I built this version, a 15" flat screen LCD was still expensive, but these days a much larger video screen and even multiple monitors for side views is possible. Cabin construction is incredibly light weight, inexpensive, and simple. The floor is made from 1/2" plywood, and the walls and ceiling are made from 1/8" 4' x 8' tempered hardboard.

Hardboard is available from all major lumber stores for

Parts List (of sorts)

- (4) 1K 1/2 watt resistor
- (4) 10K 1/2 watt resistor
- (2) 2.5K potentiometers (highest quality possible)
- (4) IRF540 MOSFETs
- 7805 voltage regulator (to provide 5V current to feedback potentiometers)
- 24V DC power supply (select amp rating to suit your chosen solenoid valves)
- Weeder Technologies digital I/O module WTDIO-M
- Weeder Technologies analog input module WTADC-M

Note: My Virtual Flyer's Weeder boards connected to the computer's serial port.

If your computer lacks a serial port, a USB to serial adapter cable has been tested with the Virtual Flyer and will work.

I do not have experience with newer USB based boards or if they can be accessed from Quick Basic.

- (4) 24V solenoid valves (ideally 1/4" 150 PSI or higher)
- Note: Larger air cylinders can run at lower pressures, so lower PSI air valves than previously mentioned can also work in that instance.

Parts Sources:

An extraordinary, inexpensive source for surplus solenoid valves and cylinders is **www.surpluscenter.com**.

For hoses and fittings including cylinder flow controls (needle valves), go to **<http://www.clippard.com/>**.

Clippard's components are the highest quality.

I strongly recommend Clippard's needle valves such as the MNV-3P to serve as your cylinder flow controls.

about \$9 a sheet, and bends easily into many shapes.

By gluing small wood blocks at all corners and along seams as needed, the your cabin's hardboard panels can literally screw together. All seams are then reinforced, and gaps are filled using a simple construction adhesive such as Liquid Nails™. This design withstood many years on the road, and worked perfectly for riders up to 220 pounds.

If you're not into carpentry, no problem! **Figure 5** shows an even simpler cabin made using standard PVC pipe and fittings. By simply covering such a PVC frame with fabric or any other thin material, a very simple "hood" can be made for quick experimenting. Ultimately,

your cabin should be designed to fit your needs.

Mounting to a Pedestal

For this particular simulator, I chose a pedestal mounting system (**Figure 6**). This is nothing more than a scrap automotive universal joint (as you might find on a drive shaft) from a junkyard. Usually, these cost little or nothing.

I hired a local welder for a few dollars to cut and weld this shaft to a large metal plate. Looking again at **Figure 2**, note that the pedestal is *not* centered under the cabin. It should be mounted so that it is far enough back on the cabin that *some weight will always be on the front pitch cylinder*. By keeping the cylinder under constant load, you will always have control of your platform's position.

The same is true for your side roll axis. As seen in **Figure 7**, the pedestal mounts slightly behind and aft of the rider and simulator's center of gravity to keep both the pitch and roll cylinders under constant load. You will also notice in the photo that I had two chains and springs attached to keep an additional load on each cylinder at all angles.

You may or may not find this necessary, but since I wanted a wild ride, I added the springs so that the tilt of the cabin would be as rapid and wide as possible on the return stroke of each cylinder.

The Electronics

Thanks to Weeder Tech — whose ads I found in *Nuts & Volts* years ago — electronic connection of the original simulator to the computer was extremely simple. Let's briefly talk about what the electronics do.

The Virtual Flyer is a complete closed-loop system, which is to say that the computer running the simulation monitors the exact position of the simulator platform and adjusts that position to match events on the screen approximately 20 times each second.

In order to accomplish this, your computer simply needs an input/output module capable of opening and closing the four air valves that move the simulator. It also needs to read the position of the platform via two feedback potentiometers.

Figure 6 shows the two feedback potentiometers on the base of the simulator. Attached to each potentiometer is a heavy armature and a length of chain that attaches to the base of the cabin.

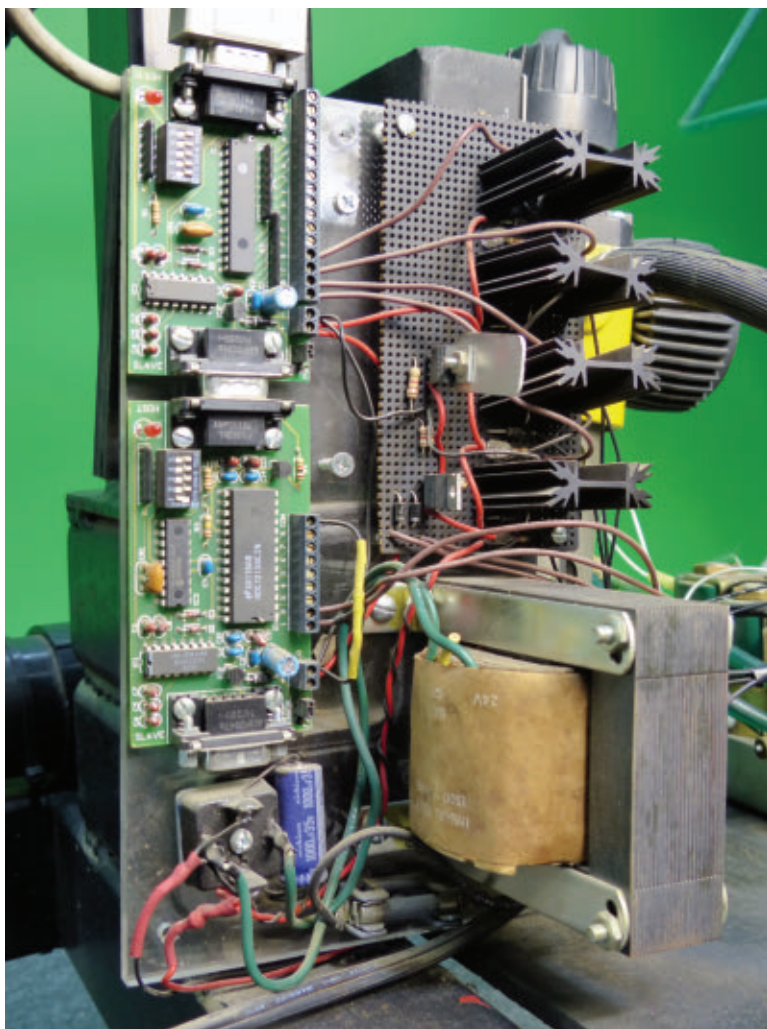
One chain is attached to register pitch



■ **FIGURE 6.**



■ **FIGURE 7.**



■ FIGURE 8.

movement and one is to detect roll position. As the cabin moves, the chains raise and lower the potentiometer arms and the values are captured by the computer. These days, many modules exist that are capable of performing both functions and connecting via USB or other simple means to your computer.

In addition, many terrific microcontrollers could probably do the whole job without burdening the computer at all. So, you are not necessarily tied to the Weeder control boards I used (though the software I wrote would have to be modified for your board of choice). This is a key place where you may enjoy modernizing the design.

For the original Flyer, I used Weeder Tech's Digital I/O Module (WTDIO-M) to operate the solenoid valves, and Weeder Tech's analog input module to read the values from the two potentiometers.

Figure 8 shows the entire circuit board connected. As you can see, electronic control for the simulator is simply two Weeder boards and a set of IFR 540 MOSFETS which handle the load when opening and closing the four air valves.

Figure 9 shows how the IFR 540s and potentiometers are connected between the Weeder board and the air valves. (Each of the pitch and roll connections are connected to the same IFR 540 circuit shown for pitch 1.)

Software

Three pieces of software are needed for your simulator to fly:



PBP3
PICBASIC PRO™ Compiler

Experimenter - \$49.95

Silver Edition - \$119.95

Gold Edition - \$269.95

The industry-standard BASIC compiler for Microchip PIC® microcontrollers.

Multi-Seat Licensing
for Educational Institutions

Upgrade from
PICBASIC™ Compiler (PBC)

Download a FREE trial version now.

www.PBP3.com

microEngineering Labs, Inc.

www.melabs.com

888-316-1753

PICBASIC and PICBASIC PRO are trademarks of Microchip Technology Inc. in the USA and other countries. PIC is a registered trademark of Microchip Technology Inc. in the USA and other countries.

MakerPlot ^{Part 1}

The DIY Software Kit

By John Gavlik
and Martin Hebel

Post comments on this article and find any associated files
and/or downloads at [www.nutsvolts.com/
index.php?/magazine/article/april2014_MakerPlot](http://www.nutsvolts.com/index.php?/magazine/article/april2014_MakerPlot).

This month, we're going to introduce you to MakerPlot's best kept secret – the Logs(Debug) Immediate window. It's really not a secret, just seems so when it's mostly overshadowed by all the other graphical functions that MakerPlot is known for. It's buried behind a small logbook icon (**Figure 1**) on the toolbar, so it's easy to miss. As they say, big things come in small packages, and the Logs(Debug) Immediate window is one of them.

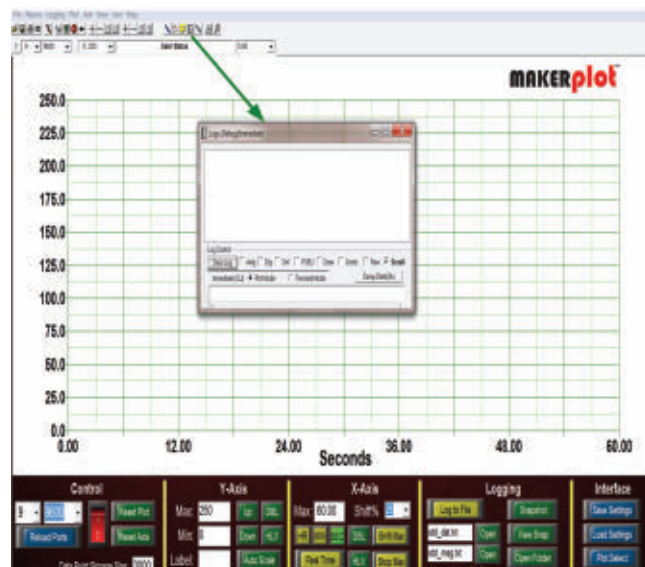
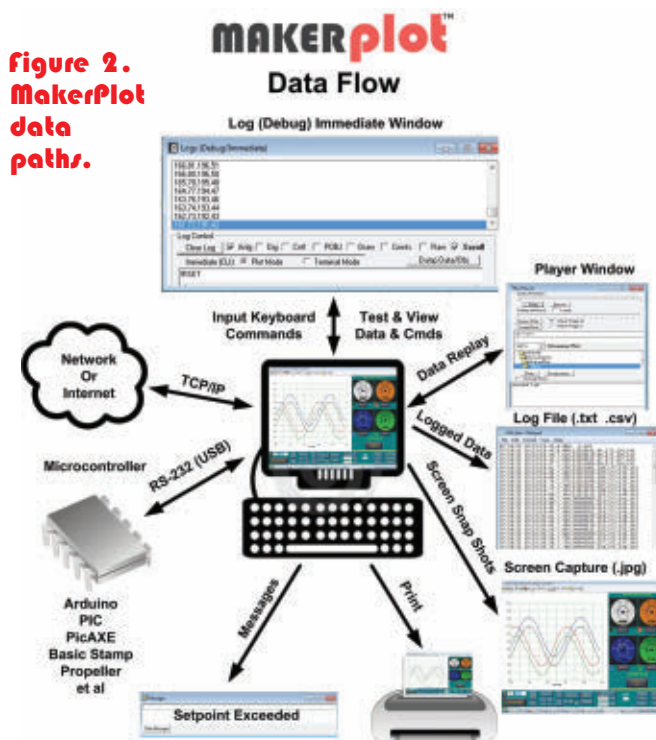


Figure 1 . The logs(Debug) Immediate window.

If you've been following this series, you may remember in Part 1 where we devoted nearly the entire article to how MakerPlot can be a debug tool for your micro's code. This time, we're going to get into the thick of things and show you how the **Logs(Debug) Immediate** window can make that happen. If you haven't already done so, you can download a free 30 day trial copy of MakerPlot from www.makerplot.com to follow along. (Be sure to receive your \$5 discount off the regular MakerPlot price by using this discount code if you decide to order: NVMP092713.) Let's get going.

MakerPlot Data Paths

Data in and out of MakerPlot can come from many different sources and directions; **Figure 2** is an illustration of this. What's important to note is that the majority of the data flow from all these diverse directions are captured in the **Logs(Debug) Immediate** window at the top of **Figure 2**. This means that anything coming into and going out of MakerPlot can be seen, stored, and analyzed. This not only includes the analog and digital data that the code sends from your microcontroller, but also




```
// MakerPlot Arduino Bi-Directional Control - Part 2
// This sketch is in the public domain

const int analogInPin = A0; // Analog input pin that the potentiometer is attached to
const int LEDpin = 9; // Analog output pin that the LED is attached to
const int SW1pin = 2; // Pushbutton switch, raise setpoint
const int SW2pin = 3; // Pushbutton switch, lower setpoint

int sensorValue = 0; // value read from the pot
int setPoint = 100; // Initial value of setPoint
int SW1state = 0; // Store state of SW1
int SW2state = 0; // Store state of SW2
int LEDstate = 0; // Store state of LED

void setup() {
  // configure hardware
  pinMode(SW1pin, INPUT_PULLUP); // Enable pull-ups on switches
  pinMode(SW2pin, INPUT_PULLUP);
  pinMode(LEDpin, OUTPUT); // set LED to be an output pin
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);

  delay(2000); // allow connection to stabilize
  Serial.println(); // send in case garbage in queue
  Serial.println("IRSET"); // Reset the plot
  Serial.println("IO butMin=1"); // Set butMin object to 1, ON
  Serial.println("IO butMin.Run"); // Run event code button to configure minutes, turn other 2 off
  Serial.println("IO butMax=2"); // Set text box for maximum time to 2 minutes
  Serial.println("IO butXMax.Run"); // Run text box event code to update time
  // clear constant drawings and place constant text on plot
  Serial.println("ICLRCR(CR)@TEXT 30A,105A,1.5A,(Blue),Controlled from Arduino!");

  delay(3000); // delay 3 seconds

  Serial.println("IO LED0=1"); // turn ON LED0 on the interface
  Serial.println("IO SW0=1"); // turn ON SW0 on the interface
  Serial.println("IBELL"); // sound PC Bell
  delay(1000); // delay 1 second

  Serial.println("IO LED1=1"); // turn ON LED1 on the interface
  Serial.println("IO SW1=1"); // turn ON SW1 on the interface
  Serial.println("IBELL"); // sound PC Bell
  delay(1000); // delay 1 second

  Serial.println("IO LED2=1"); // turn ON LED2 on the interface
  Serial.println("IO SW2=1"); // turn ON SW2 on the interface
  Serial.println("IBELL"); // sound PC Bell
  delay(1000); // delay 1 second

  Serial.println("IO LED3=1"); // turn ON LED3 on the interface
  Serial.println("IO SW3=1"); // turn ON SW3 on the interface
  Serial.println("IBELL"); // sound PC Bell
  delay(1000); // delay 1 second

  Serial.println("IO LED=0"); // turn OFF all LEDs
  Serial.println("IO SW=0"); // turn OFF all switches
  Serial.println("IBELL"); // sound PC Bell
  delay(1000); // delay 1 second
}
```

Figure 3. Part 1 of bi-directional setpoint sketch.

```
void loop() {
  Serial.println("TREAD(Slider)"); // read the slider
  setPoint = Serial.parseInt(); // change the setPoint to reflect it

  sensorValue = analogRead(analogInPin); // get pot voltage into sensorValue

  // Check pot value against setpoint, if above light LED
  // And take snapshot is sensorValue > setPoint

  if (sensorValue > setPoint)
  {
    if (LEDstate == 0) // take snapshot if above setpoint,
                      // just once each crossing (if LEDstate was off)
    {
      Serial.println("IO butSnap=1"); // Turn on Snapshot button
      Serial.println("IO butSnap.Run"); // Run snapshot event code
      Serial.println("IO butSnap=0"); // Turn off snapshot button
    }
    LEDstate = 1;
  }

  else
    LEDstate = 0;
  digitalWrite(LEDpin, LEDstate);

  // Check states of pushbuttons, if pressed change setpoint up or down
  SW1state = digitalRead(SW1pin);
  if (SW1state == 0)
  {
    setPoint++; // increment setPoint by 1
    Serial.print("IO Slider ="); // send to Makerplot to adjust slider
    Serial.println(setPoint); // based on new setPoint
  }

  SW2state = digitalRead(SW2pin);
  if (SW2state == 0)
  {
    setPoint--; // decrement setPoint by 1
    Serial.print("IO Slider ="); // send to Makerplot to adjust slider
    Serial.println(setPoint); // based on new setPoint
  }

  // print the analog values formatted for MakerPlot
  Serial.print(sensorValue); // send 1st value
  Serial.print(","); // send comma delimiter
  Serial.println(setPoint); // send 2nd value with carriage return

  // print the digital values formatted for MakerPlot
  Serial.print("%"); // send binary indicator
  Serial.print(SW1state); // send 1/0 for SW1
  Serial.print(SW2state); // send 1/0 for SW2
  Serial.println(LEDstate); // send 1/0 for LED with carriage return

  // wait 100 milliseconds before the next loop
  delay(100);
}
```

Figure 4. Part 2 of bi-directional setpoint sketch.

instructions for MakerPlot that come in alongside this data. This was shown to you in the previous articles on bi-directional control, but what wasn't shown was how the **Logs(Debug) Immediate** window operates to capture and display that data. Let's do that now.

Capturing MakerPlot Instructions

We're going to continue with the bi-directional code (Figure 3) that was used in Part 6 to first show how the **Logs(Debug) Immediate** window displays the MakerPlot instructions that are transmitted from the Arduino to: 1) reset MakerPlot; 2) adjust the time scale; and 3) turn the four LEDs and toggle switches ON then OFF. Take a look at the code in Figure 3 to see what that means; Figure 5 shows how it looks as it's happening in the **Logs(Debug) Immediate** window.

If you compare the sketch against the **Logs(Debug) Immediate** window, you'll see the one for one correspondence between plotted and recorded data. In order to see this information, you'll need to have the

POBJ and **Scroll** boxes checked (red ovals). What's interesting to note is that you'll have a record of what was sent by your microcontroller's code via the serial link.

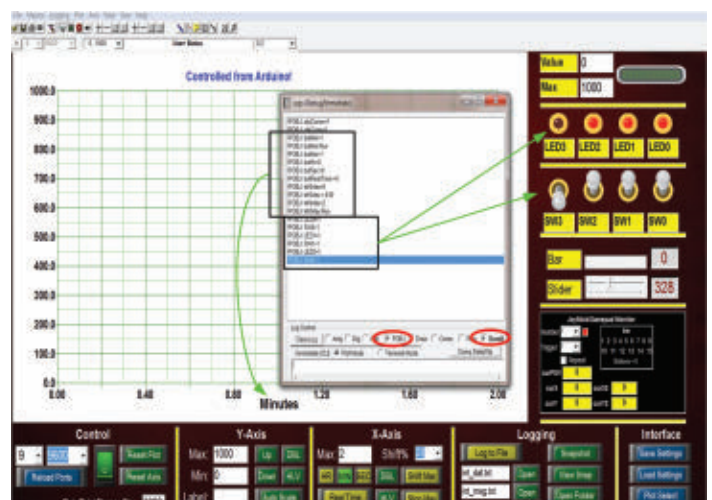


Figure 5. MakerPlot instructions.

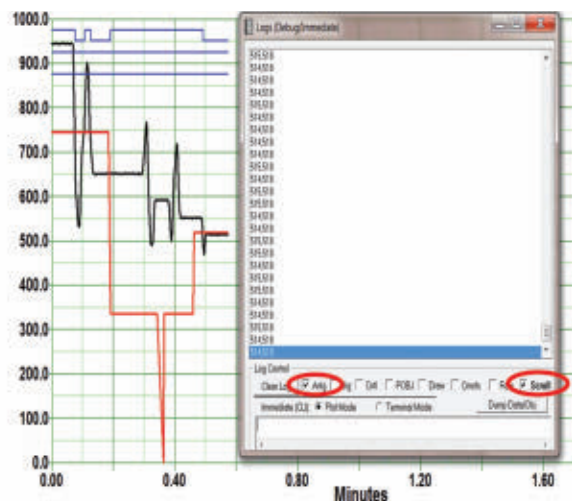


Figure 6. Analog data display.

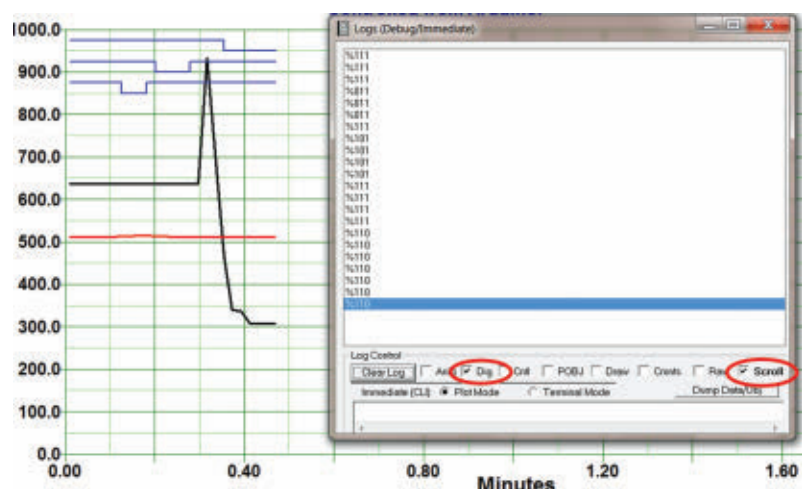


Figure 7. Digital data display.

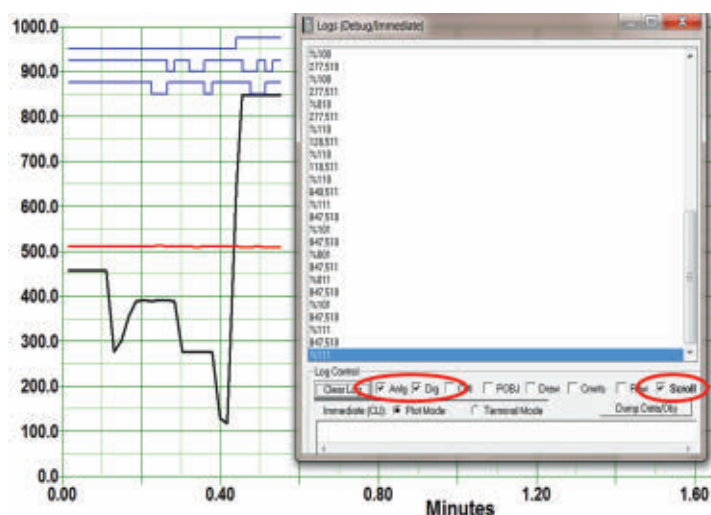


Figure 8. Mixed analog and digital data display.

While this particular code works, yours may not the first time, so the **Logs(Debug) Immediate** window is a good way to check to see exactly what is being received by MakerPlot for code debugging purposes. Now, let's get to the analog and digital data part.

Capturing Analog and Digital Data

Figure 4 is the listing for the loop part of the sketch that outputs analog and digital data, along with more instructions to MakerPlot. Recall from Parts 5 and 6 that this is an interactive setpoint application and that the analog data consists of the potentiometer and setpoint values. The digital values consist of the two pushbutton switches and the above (1) or below (0) setpoint crossing level. Let's check out what they look like in the **Logs(Debug) Immediate** window.

Beginning with the analog data, Figure 6 illustrates how MakerPlot receives these two values. To view analog data, the **Anlg** and **Scroll** boxes (red ovals) need to be checked. As you can

see, there are two sets of numbers separated by a comma. The first is the potentiometer value (the black plot), followed by the setpoint value (the red plot). So, as the plotted pot and setpoint values change, you'll see these values change in the **Logs(Debug) Immediate** window along with them.

To display digital data, you'll need to uncheck **Anlg** and check the **Dig** box. The result is in Figure 7 where the three digital values are displayed as 1s and 0s preceded by the percent (%) sign. As the SW1 and SW2 pushbuttons are pressed, the data changes. If the potentiometer level goes above or below the analog setpoint, the digital setpoint changes from 1 to 0, respectively (the right-most number). This is evident on the top three plot lines, as well. You can see both the analog and

digital values together by simply checking both the **Anlg** and **Dig** boxes. Figure 8 shows how this looks. The take-away from all of this is that the **Logs(Debug) Immediate** window provides you with a double-check on what your micro is sending to MakerPlot, in addition to the regular plotted lines.

Manually Entering Instructions and Data

Now, let's go the other way and show you how to manually enter both instructions and data into MakerPlot using the **Logs(Debug) Immediate** window. This is in contrast to the instructions and data coming from the micro via the serial link; now they're going to come into MakerPlot through the CLI window – the rectangular text box at the bottom. To do this, you'll

need to click the green rocker switch until it turns red to break the serial connection from your micro, just to keep the micro's data from interfering with the manually input data. Then, you'll need to click on the small **Plot** icon on the toolbar (the one that looks like a computer monitor) to get plotting going. Finally, bring up the **Logs(Debug) Immediate** window and click the **Clear Log** button to start fresh.

To manually enter commands and data, simply key them into the rectangular area at the bottom of the window and push the Enter key (**Figure 9**). We've started with the **!RSET** instruction in order to clear the plot area and to reset the plot to time zero. We followed this with six groups of analog data, with each group having three values. You can see the reset instruction and the six analog data groups in the main window, along with the matching plotted lines that are created by the analog data.

Each line has its own color to identify it, with black as channel 0; red as channel 1; and blue as channel 2. You can have up to 10 analog channels — each with their own color. Of course, you can change those plot line colors, but that's a subject for another article.

The point about entering analog data manually or in code is that MakerPlot treats any string of numerical ASCII characters that are separated by a comma and terminated with a carriage return as analog data; no other prefixes or suffixes are necessary.

It's a little different for digital data. With digital data, you need to prefix the 1 and 0 ASCII string with a percent (%) character followed by the data itself, then a carriage return. **Figure 10** is such an example. Again, we started with the **!RSET** instruction and keyed in five sets of digital data, although we could have gone up to 32. You can mix analog and digital data along with instructions and messages together. That's the beauty of this simple yet powerful debugging tool.

Terminal Mode

Up until now, we've been in Plot mode. For this next example, we've switched to Terminal mode (**Figure 11**). The display changes to yellow to alert you to the changed display mode. This is yet another way to see what's coming in from your micro's serial link. Here, you can see what MakerPlot sees as the **!READ(Slider)** instruction is followed by the analog setpoint and pot values, then the digital values for the pushbutton switches and the setpoint crossing level.

Video

To get a better idea of how the **Logs(Debug)**

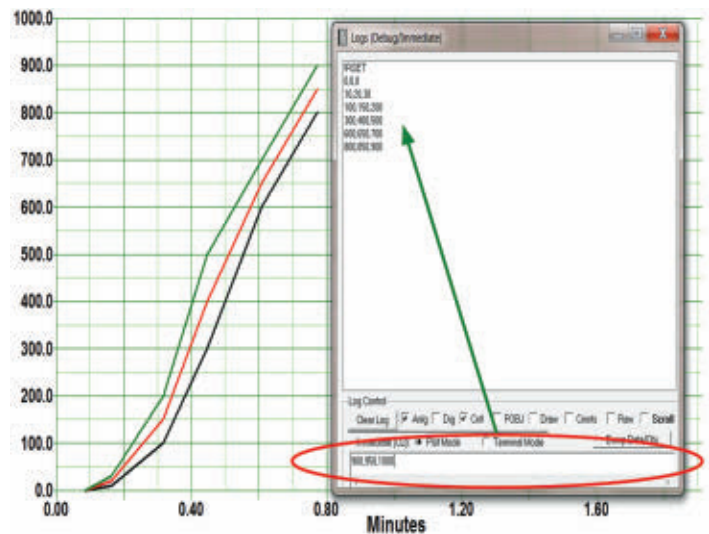


Figure 9. Manually entering instructions and analog data.

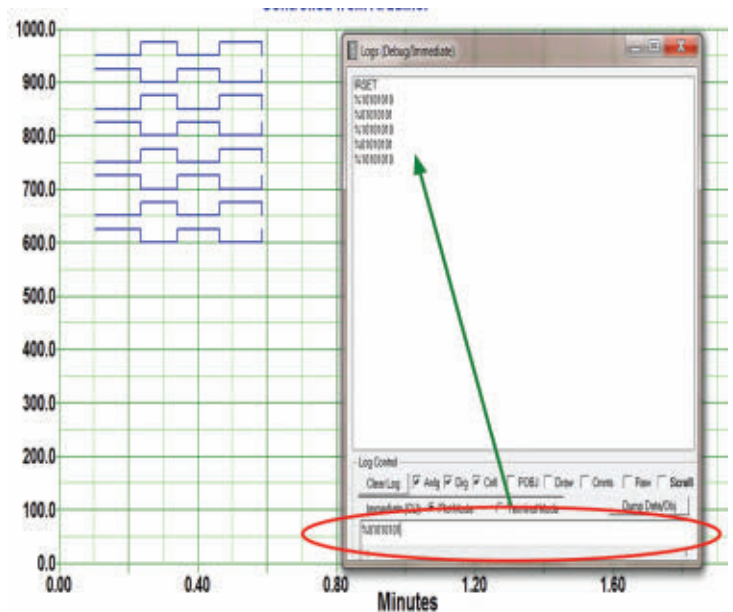


Figure 10. Manually entering instructions and digital data.

Immediate window works, go to the MakerPlot website and follow this path **Basic Plotting Video → Logs(Debug) Immediate Tab**. Here, you can see what happens in the window with live data coming in; it will give you a much better handle on how it can work for you.

Conclusion

To sum up, you've been introduced to the **Logs(Debug) Immediate** window for two reasons. The

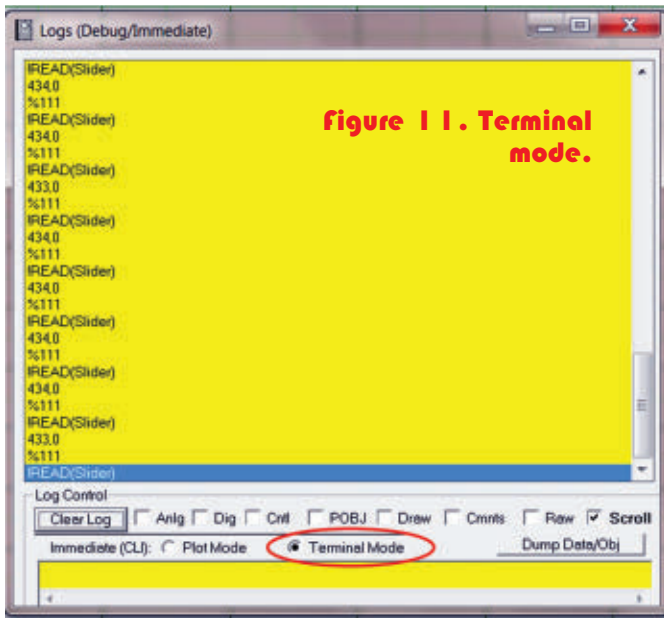


Figure 11. Terminal mode.

alongside of it, you can acquire a more quantified idea of what's happening on the micro's side in order to find a coding error. The other reason we introduced you to the **Logs(Debug) Immediate** window is to form a basis for the next couple of articles on customizing MakerPlot; this is where we'll show you how to build your own Interface screen from scratch (or nearly so).

We'd like you to think of MakerPlot as the software equivalent to a hardware front panel. If you were to build a hardware panel for your micro's application, your "kit" would include pushbuttons, switches, meters, and maybe a graphic display. All of this is time-consuming, difficult to implement, and probably expensive — not to mention cast in stone once it's all done.

The name MakerPlot came into existence once it dawned on us that what it really does is act like a software kit where you can graphically assemble any kind of front panel you want in order to display and control your data. That's the real power within MakerPlot; that is, you can customize it to your own needs, and that's what we'll get into next time.

That's all for now, so just remember: **Got Data – MakerPlot It! NV**

primary reason was to show how you use it to view the data and instructions coming from your micro into MakerPlot, mainly to debug your code. When you use the **Logs(Debug) Immediate** window with the plotted data

The Convenient All-in-One Solution for Custom-Designed Front Panels & Enclosures



FREE Software



ONLY \$90.24
with custom
logo engraving

You design it
to your specifications using
our FREE CAD software,
Front Panel Designer

We machine it
and ship to you a
professionally finished product,
no minimum quantity required

- Cost effective prototypes and production runs with no setup charges
- Powder-coated and anodized finishes in various colors
- Select from aluminum, acrylic or provide your own material
- Standard lead time in 5 days or express manufacturing in 3 or 1 days



FrontPanelExpress.com
1(800)FPE-9060

WORLD'S MOST VERSATILE CIRCUIT BOARD HOLDERS



Model 324

Our Circuit Board Holders add versatility & precision to your DIY electronics project. Solder, assemble & organize with ease.

VISIT US ON
 

MONTHLY CONTEST
Visit us on Facebook® to post a photo of your creative PanaVise project for a chance to win a PanaVise prize package.



Model 201



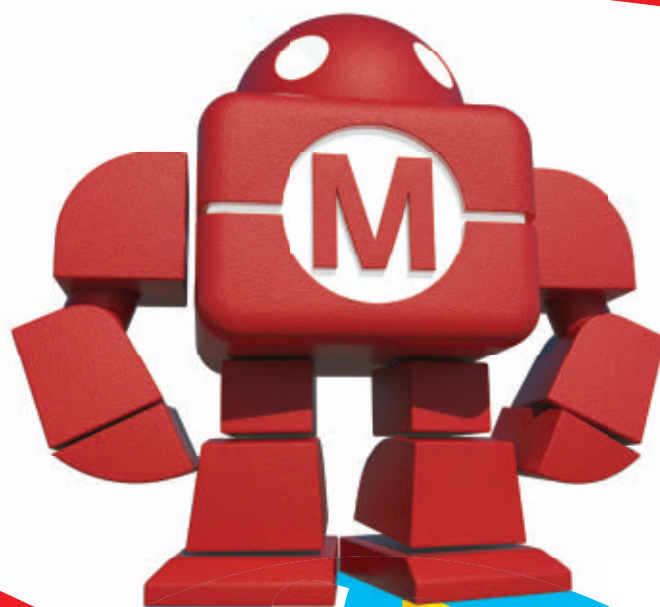
PANA VISE®
Innovative Holding Solutions

7540 Colbert Drive • Reno • Nevada 89511 | (800) 759-7535 | www.PanaVise.com

Maker Faire

YEAR OF THE MAKER

CELEBRATING COMMUNITY
AROUND THE WORLD



BAY AREA

NEW YORK

KANSAS CITY

OSLO

UK

ROME

TOKYO

SHENZHEN

**GREATEST SHOW
& TELL ON EARTH**

9th Annual **BAY AREA**
MAY 17+18

5th Annual **NEW YORK**
SEPT 20+21

makerfaire.com
Brought to you by **MAKE** magazine

Software-Defined Radios Everywhere

In case you haven't noticed, virtually all radios today — cell phones and Wi-Fi WLAN, for example — are software-defined radios (SDRs). More and more, the functions of a radio are gradually moving from hardware to software. Furthermore, with ever decreasing semiconductor sizes, chip companies can put more circuits and functions on a chip that operates at higher and higher frequencies. Today, it is no problem to put most of a radio on a single chip. Two recently announced integrated circuits really point this out. Here is a look at the latest in SDR RF chips.

WHAT IS AN SDR?

First — as a quickie review — a software-defined radio is a digital radio where some or most of the operations of a transmitter or receiver are performed digitally with mathematical algorithms. These digital operations can take place in a digital signal processor (DSP) or in a field programmable gate array (FPGA). The most common operations are filtering, mixing, modulation, and demodulation. Amplification is still an analog operation, however.

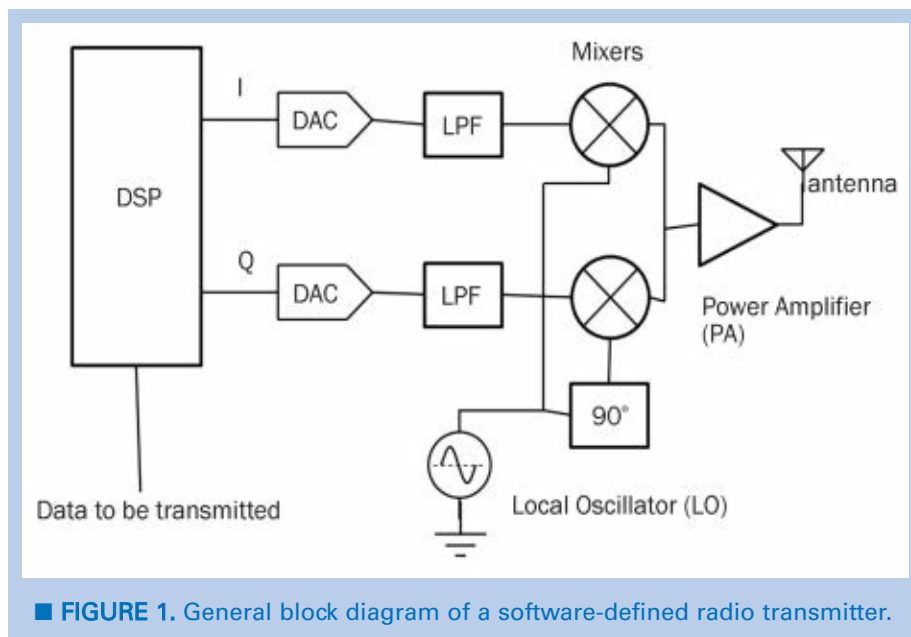
An SDR transmitter would first put any information (voice, video, etc.) to be sent into digital form with an ADC (analog-to-digital converter) and send it to the DSP where the modulation is applied. Refer to **Figure 1**. The DSP generates two digital outputs called the in-phase (I) signal and the quadrature (Q) signal. These signals

are called the baseband (BB) signals.

The signals are next sent to DACs (digital-to-analog converters) where the analog radio signals are developed then filtered by low pass filters (LPF). The signals are then upconverted by a pair of mixers. The mixers are fed with a local oscillator (LO) signal set to the final transmit frequency. One LO signal is shifted 90 degrees from the other. The mixer outputs are added together then sent to a power amplifier (PA) and the antenna.

Note: The reason for the I and Q signals is that both are needed at the receiver to recover the original modulating data. The I and Q signals provide the amplitude and phase information needed by the mathematical algorithms for demodulation.

The ideal SDR receiver would be an antenna connected to a low noise amplifier (LNA) and then to an ADC. Check out **Figure 2**. An input filter is

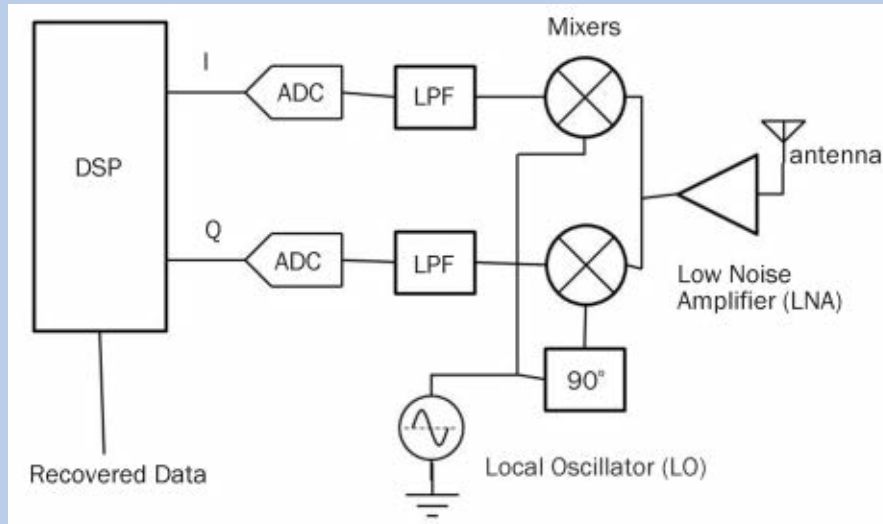


■ **FIGURE 1.** General block diagram of a software-defined radio transmitter.

ANALOG DEVICES AD9361

Figure 1 shows the AD9361. It is actually two complete transceivers in one: two receivers and two transmitters. It is designed for those wireless services that use MIMO. MIMO is multiple input multiple output – a technique for dividing a high speed digital signal into two separate paths and transmitting them simultaneously on the same frequency using special coding methods. MIMO has the effect of mitigating the fading and reflections normally encountered by high frequency signals. It also multiplies the data rate by the number of transmitters and receivers used. The AD9361 is good for 2 by 2 (or 2x2) MIMO.

The operational frequency range of the chip is 70 MHz to 6 GHz – a huge range – making it suitable for



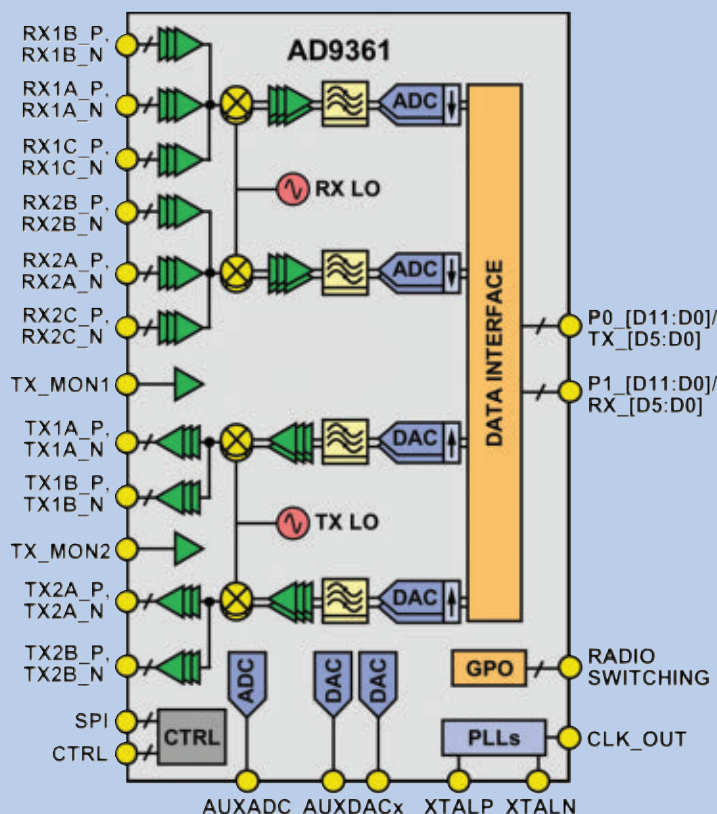
■ **FIGURE 2.** A typical direct conversion software-defined radio receiver.

normally included to define the received frequencies. The LNA boosts the signal level to that needed by the ADC. The LNA output is sent to a pair of mixers driven by two local oscillator signals shifted 90 degrees, but set to the input signal frequency. The mixer outputs are filtered in low pass filters to create the I and Q baseband signals. The ADCs digitize the received I and Q signals and send them to the DSP or FPGA where filtering and demodulation (and perhaps other operations) are performed. The baseband signals are recovered. The DSP output may then go to a DAC to recover the original analog modulating signal like voice or video.

There are varying degrees of SDRs. Some may use more external RF circuitry like filters or mixers. Some radios have all of the analog radio functions on one chip and all of the DSP on another chip. The ADCs and DACs may be on either chip depending upon the application.

SDRs are everywhere. They are in cell phones, cellular basestations, and Wi-Fi empowered devices like laptops, tablets, and routers. TV sets are SDRs. A recent development is programmable SDR chips that can operate over a wide range of frequencies in different modes. Two

good examples are the ICs to be discussed next. Both represent the RF front-end of an SDR, and both are fully programmable.



■ **FIGURE 3.** Simplified diagram of the Analog Devices AD9361 programmable SDR transceiver.

many different wireless applications. It targets cellular basestations and the forthcoming small cells movement since both use MIMO. It can also be used for Wi-Fi and WiMAX radios. Any commercial or military radio is a candidate to use this chip.

Take a look at the diagram in **Figure 3**. The two receivers are at the top. Each has three LNA amplifiers on

In case you haven't noticed, virtually all radios today – cell phones and Wi-Fi WLAN, for example – are software-defined radios.

the left to amplify the input signals which come from the antenna, probably by way of external filters. The inputs are selected one at a time and sent to the mixer. The other input to the mixer is the receiver local oscillator, or RX LO. The LO is a phase-locked loop (PLL) frequency synthesizer that can be set to any value in the frequency range indicated earlier.

The receivers are of the direct conversion type where the LO is set to equal the incoming signal

frequency. The mixer output is then the original baseband signal. It is filtered then sent to the 12-bit ADC. The filter bandwidth can be set to any value in the 200 kHz to 56 MHz range, depending upon the application.

The ADC outputs the digital version of the signal to the output pins via the data interface. The outputs are parallel digital lines labeled P1 (D11-D6)/RX (D5-D0). This digital signal is then sent to a DSP or FPGA where it is demodulated and otherwise processed for final use. Keep in mind that there are two identical

independent receivers, and both have automatic gain control.

The two transmitters (TX) are at the bottom of the diagram in **Figure 3**. They take the digital data from a DSP, FPGA, or other source, and apply it to the input pins labeled P0 (D11-D6)/TX (D5-D0) and the data interface. The digital inputs are then converted to analog by the 12-bit DACs. These analog signals are filtered and sent to the upconverting mixers along with the transmit local oscillator (TX LO).

The transmitter is of the direct

conversion type that puts the modulating information right on the desired output frequency set by the transmitter PLL synthesizer. The final output goes to some internal RF power amplifiers used to boost signal level. The transmitter outputs go to external channel selection filters and a power amplifier before going to the antenna.

The whole chip is programmable. Using digital control words from an external processor, you can set transmit and receive frequencies, filter bandwidths, and input/output selection. The programming interface is the familiar serial peripheral interface (SPI).

The AD9361 is contained in a 144-pin chip scale package ball grid array (CSP_BGA) that is only 10 mm x 10 mm in size. The chip operates from three supplies, or 3.3, 1.8, and 1.3 volts. All you need to make a complete radio is the external DSP/FPGA and any necessary tuning filters.

LIME MICROSYSTEMS LMS7002M

The Lime Microsystems LMS7002M is similar to the AD9361 in that it is a dual programmable transceiver designed for 2x2 MIMO applications. It targets cellular basestations, small cells, Wi-Fi, WiMAX, and other SDR radio applications. It covers the frequency range from 50 MHz to 3.8 GHz.

Figure 4 shows a simplified block diagram of the device. The upper section is the dual transmitter. The digital signal to be transmitted is developed in an external DSP or FPGA and sent by the parallel LimeLight interface that is compatible with the popular JESD207 interface. There are two signals: the I or in-phase signal; and the Q or quadrature signal. All DSP operations work with these two signals.

Some undefined DSP is performed, then they go to the I and

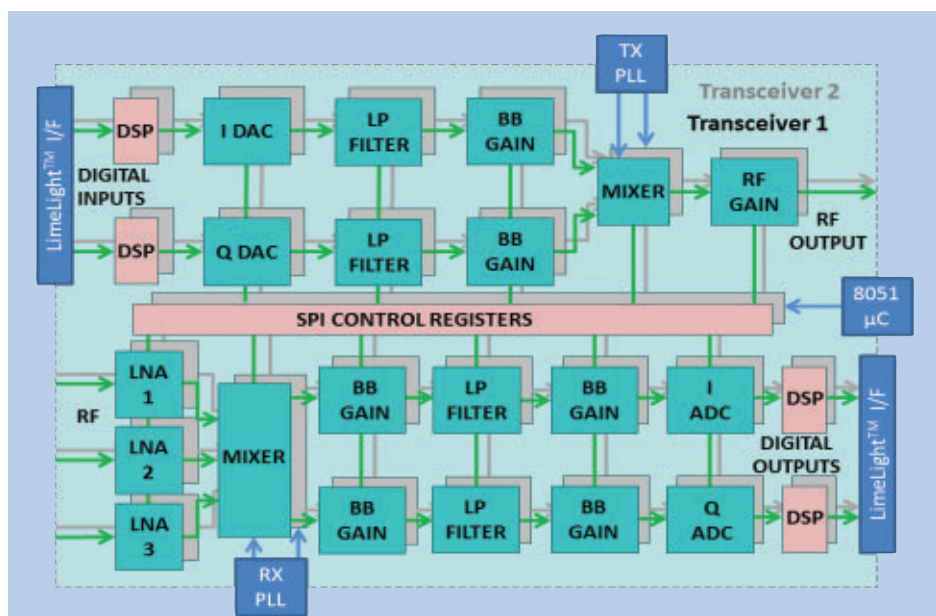


FIGURE 4. Simplified diagram of the Lime Microsystems LMS7002M programmable SDR transceiver.

Q DACs where they become analog signals. They are filtered to set the bandwidth somewhere in the 1.5 to 28 MHz range. The signals are amplified and fed to the mixer. The transmitter is the direct conversion type that upconverts the signals to the desired output frequency. An internal PLL synthesizer drives the mixer. The output is then amplified to +3 dBm that is sufficient to drive an external power amplifier.

The dual receivers are at the bottom of **Figure 4**. Three LNAs accept inputs from the antenna by way of separate tuned circuits or filters. The mixer then downconverts these signals to baseband. Again, a direct conversion architecture is used where the receiver PLL is set to the receive frequency. The baseband signals are amplified and low pass filtered, amplified again, and fed to the I and Q ADCs. Some DSP operations occur on-chip before the digital signals go to the LimeLight interface bus for transport to the external DSP/FPGA.

As with the AD9361, the LMS7002M is fully programmable. It uses the SPI interface to send digital codes to set the operating frequencies, bandwidths, gains, and other parameters. The LMS7002M also has an onboard 8051 microcontroller to aid in the programming and configuration of the chip.

The LMS7002M comes in an 11.5 mm x 11.5 mm QFN package with 261 pins. It operates from a 1.8 volt supply.

Both of the radio chips make designing complex wireless devices relatively fast and easy. Because there is so much circuitry inside, it lessens the number of discrete external components needed, while cutting costs and minimizing printed circuit board space.

Radio design with chips like these becomes more of a programming exercise than hardware design. That is the nature of software-defined radios. **NV**



Subscribe to **SERVO**

It's All About Robots

Do You Like Call 877 525-2539

- ✓ Designing Robots
- ✓ Building Robots
- ✓ Programming Robots
- ✓ Robot Competitions
- ✓ All Of The Above

Only \$26.95

www.servomagazine.com

\$51^{For 3} PCBs

FREE Layout Software!

FREE Schematic Software!



- 01 DOWNLOAD our free CAD software
- 02 DESIGN your two or four layer PC board
- 03 SEND us your design with just a click
- 04 RECEIVE top quality boards in just days

expresspcb.com

The Arduino Classroom

Arduino 101 — Chapter 4: Digital Input ... Pushbuttons

Computers use LEDs to tell us something just as often as we use a pushbutton to tell the computer something. An alarm clock buzzes and the LEDs tell us what time it is. We then push a button to tell the alarm clock that we want to sleep another few minutes (snooze button) or maybe we're ready to get up, so we push the alarm off button. These two buttons tell the computer two different things. The snooze button tells it to turn off the alarm and set a new alarm for some time in the future. The alarm off button tells the computer to turn the alarm off and reset the new alarm time for 24 hours in the future. The alarm clock has a built-in microcontroller (not unlike the one on the Arduino) that turns the LEDs on and off to show you the time, and reads the buttons to learn what you want it to do next. In this chapter, you will learn how to design circuits using pushbuttons and how to utilize them to get user input with Arduino software that will let your system take actions when a button is pressed.

Before we get into that, let's learn another couple of Arduino C programming concepts that we'll use when testing pushbuttons. We'll look at decision-making using the *if-else* conditional flow control construct. Then, we will learn to use *do-while*, which is similar to the *while* flow control construct we saw last month. Next, we will learn how to use the Arduino function *millis()* to do some event timing. Finally, we will learn about '=' and '==' — the two C operators that surprisingly aren't equal.

We will apply this knowledge in our labs where we'll learn how to get the Arduino to detect a button push and use that information to control LED states. For our last exercise, we will bring it all together along with the *millis()* function to create an Arduino-based reaction timer.

How quick can you get your finger off a pushbutton after an LED turns on? Well, by the end of this chapter you will know!

More Decisions: *if-else*

The Arduino provides several ways for a program to make decisions. One of these is to pose the question: "*if* this is true do this, *else* do that." It examines a statement

to see if it is true. If it is true, then it does one thing; if that statement is not true, then it does something else. The question is posed in code as follows:

```
if (statement is true)
{
    // do this;
}
else
{
    // do that;
}
```

In this chapter's lab's pushbutton LED examples, we ask the question: "Is the button pushed?" which we can determine by looking at the Arduino pin the pushbutton is connected to and seeing if it is HIGH. If it is true that the pin is HIGH, then we turn the LED on. If it is not true, then we turn the LED off as follows.

First, get the pushbutton state HIGH or LOW by using the *digitalRead* function:

```
pushButtonState = digitalRead(pushButtonPin)
```

Next, we use the *if/else* statements to turn the LED on if the pushbutton state is HIGH, and turn it off if it is LOW.

[Note that this uses the `==` operator to determine if the item on the left is equal to the item on the right. We'll discuss this operation in detail in the next section.]

```
if (pushButtonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
}
else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
}
```

Later, we'll see many other examples for using *if-else* for making decisions based on the statement within the *if* parentheses. For instance, we might ask if one variable is larger than another using the `<` (less than) operator as follows:

```
if(firstVar < secondVar)
{
    doThis();
}
else
{
    doThat();
}
```

We can combine *if-else* to ask several questions about the operator:

```
if(firstVar < secondVar)
{
    doThis();
}
else if(firstVar > secondVar)
{
    doThat();
}
else
{
    doTheOther();
}
```

Another Control Loop: *do while()*

In Chapter 3, we learned about the *while()* control structure that runs the block of code that follows the *while(condition)* each time it checks the condition and finds it true. A variant on this is the *do while()* control structure which is similar to *while()* except that it — at least once — runs the code in the block that follows the *do*, regardless of the condition in the *while*. In a regular *while(condition)* loop, the following block will not be run the first time through if the condition is false; in *do while()* the block runs once regardless. In the lab exercises, we will see a situation where this makes more sense:

```
do{
    // get the state of the pushbutton
    buttonState = digitalRead(buttonPin);
}while(buttonState == HIGH);
```

This makes sense if the *buttonState* variable is initialized to zero, but may have been pressed at sometime after initialization but before this bit of code runs. If we had used *while(buttonState == HIGH)* but

hadn't yet checked the button state, then it would never run the subsequent block of code.

One Way to Time Events

The Arduino keeps track of the time for up to about 50 days (when the number rolls over). If you get the number of milliseconds when an event starts and then get the number of milliseconds after the event ends, you can subtract the start milliseconds from the end milliseconds to get the elapsed time.

We will see this used in the reaction time tester lab at the end of this chapter where we will use the *do while()* (discussed above) and the Arduino *millis()* function to get a start and end millisecond value to report the reaction time:

```
// get the start time as soon as the LED
// goes off
startTime = millis();

// wait until the subjects gets his finger off
// the button
// read the button state until it is equal
// to HIGH
do{
    // get the state of the pushbutton
    buttonState = digitalRead(buttonPin);
}while(buttonState == HIGH);

// get the end time as soon as the finger is
// off the switch
endTime = millis();

// Tell the world your reaction time
Serial.print("You took: ");
Serial.print(endTime-startTime,DEC);
Serial.println(" milliseconds.");
```

Some Equals Are More Equal Than Others

We learned that operators in C are used for arithmetic-like operations and include such things as `+`, `-`, and `<`. Of all the operators we will see, the `=` and `==` seem to give most folks trouble. The `=` operator is the **arithmetic assignment operator**; it will assign the value from the right side of the `=` sign to the variable on the left side. The `==` operator is the **comparison operator**; it is used to compare the values on either side of the operator.

If they are actually equal, then the comparison is said to be true. If they aren't equal, the comparison is said to be false (giving the operation a value of zero). So, if we want to set one variable to equal the value contained by another variable, we use the assignment operator like this:

```
// First assignment
char firstVar = 5;
char secondVar = 10;

//Second assignment
firstVar = secondVar;
```

We originally set the *firstVar* to contain the value 5 and the *secondVar* to contain the value 10. We then write a statement that assigns the value of *secondVar* to *firstVar*. After this assignment, *firstVar* contains the value 10.

What if we want to know if one variable contains the same value as another? In the example above when the variables are defined, they contain different values. However, after the assignment statement, they contain the same value.

Let's consider the situation where we want to do one thing in the program if those two values are the same, and we want to do something else if they are different. We would use the comparison operator to make that decision. For example:

```
//First assignment
char firstVar = 5;
char secondVar = 10;

// First test
if(firstVar == secondVar) doThis();
else doThat();

// Second assignment
firstVar = secondVar;

// Second test
if(firstVar == secondVar) doThis();
else doThat();
```

In the above code, the first test *if/else* statements will call *doThat()*; since the variables being compared by the *==* operator are not equal. After they become equal in the second assignment, comparing them results in calling *doThis()*.

This is a contrived example to help clarify how *=* and *==* differ. In a real program, we are unlikely to know the values of the variables being compared. That is why we are asking the *if/else* questions.

Oh, but the problems these guys cause!

What is wrong with the following statement?

```
//BAD CODE:
if(ledState = true) // set LED state to true -
bad idea
{
    // do something
}
```

This *if* evaluation will always be true because you just set the *ledState* to equal true. You really meant to ask the question 'Is *ledState* equal to true?':

```
//GOOD CODE:
if(ledState == true) // if ledState is equal
to true
{
    // do something
}
```

The confusion involved between these two operators is very common; even experienced programmers make the mistake of using *=* when they meant *==*. Just remember that if you write a program where you want to see if some value is equal to another value and the program has a bug, look for the *=* and *==* operators. Before we move on, see if you can avert World War III:

```
//The World's Last C Bug
while (1)
{
    status = GetRadarInfo();
    if (status = 1)
    LaunchMissles();
}
```

[Many thanks to 'theusch' on www.avrfreaks.net for this example that he thinks originally came from Jack Ganssle.]

So, we can fix this by changing the *if(status = 1)* to *if(status == 1)* as follows:

```
// NOT The World's Last C Bug
while (1)
{
    status = GetRadarInfo();
    if (status == 1)
        LaunchMissles();
}
```

What if we accidentally introduce another bug? Can you find the reason this 'fixed' version will also start WWII?

```
//The World's Last C Bug
while (1)
{
    status = GetRadarInfo();
    if (status == 1);
        LaunchMissles();
}
```

Yep, you got it! I stuck a semicolon after the *if(status == 1);* making it a stand-alone statement. The C compiler assumes I know what I'm doing. It does nothing with the *if* statement and then moves to the next statement which is an unqualified *LaunchMissles()*. I made this very sort of error recently. Fortunately, nobody trusts me to program nuclear systems. So, yeah. Software bugs can be tricky. Now, let's mess with some hardware.

Input Versus Output of Higher and Lower Voltages

You learned how to output higher and lower voltages to control an LED in Chapter 2 where you used the Arduino digital I/O pins in the output mode. You will now learn how to use those pins in the input mode to tell if the pin is exposed to either the higher voltage (in our case, five volts) or lower voltage (in our case, zero volts). You will learn to read the pin state (HIGH or LOW) in Arduino

software to indicate that a button is pushed or not pushed, and use that information to control actions of your system.

[ASIDE: There are many other terms we may see that express the concept of what a pin reads. Where we say that the pin state is HIGH or LOW, others may say TRUE or FALSE, 1 or 0, or Vcc or GND. We are referring both to an analog concept for the voltage on the pin and a digital concept describing the pin state.]

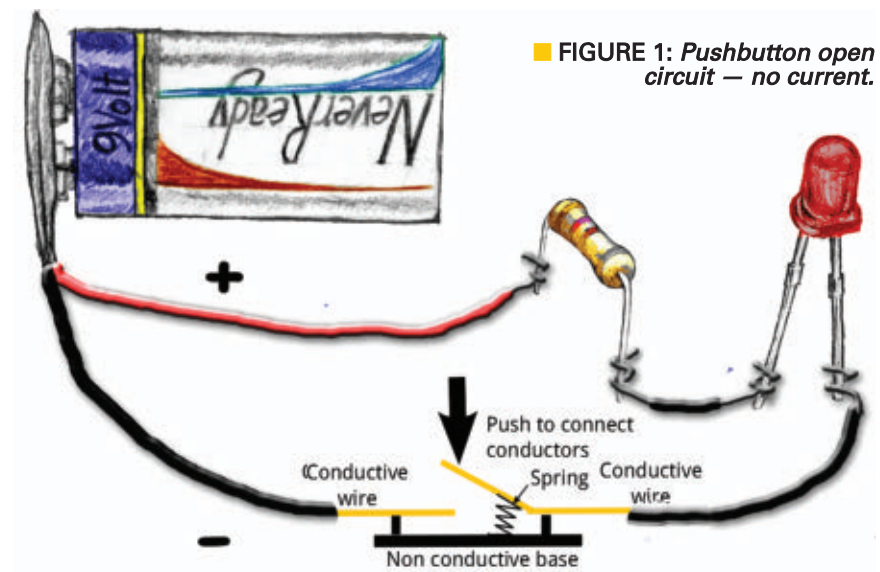
What is a Pushbutton?

There are many kinds of pushbuttons, but all serve the same purpose: to let a user connect or break an electrical circuit. Our pushbuttons are designed to break the circuit unless they are pressed, and to make the connection only when pressed. There are other buttons that are designed to keep the circuit connected unless pushed, and then to break it while pushed. Another type will toggle between connecting and unconnecting the circuit on each press.

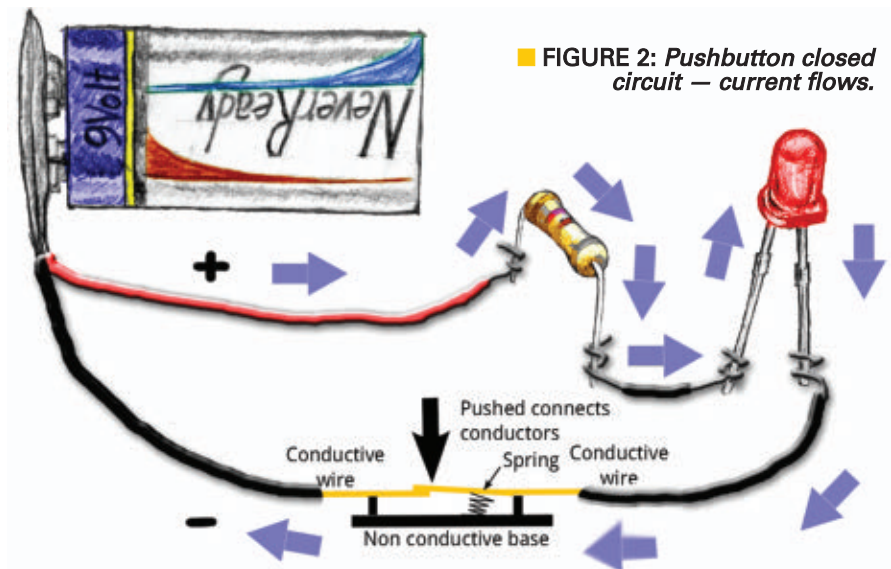
How does a pushbutton work?

In Chapter 2, we learned that a circuit is simply the complete path of a circle of conductors through which electricity can flow. If you break the path of the circle by cutting one of the conductors, the electricity will no longer flow.

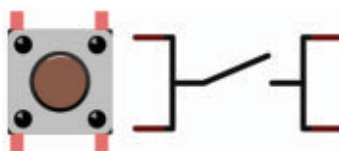
Figure 1 shows a circuit with a nine volt battery, a resistor, an LED, and a pushbutton. The pushbutton is open so that no current flows and the LED is not lit. **Figure 2** shows the same circuit with the pushbutton pressed, making the connection and allowing current to flow and light up the LED.



■ **FIGURE 1:** *Pushbutton open circuit — no current.*



■ **FIGURE 2:** *Pushbutton closed circuit — current flows.*

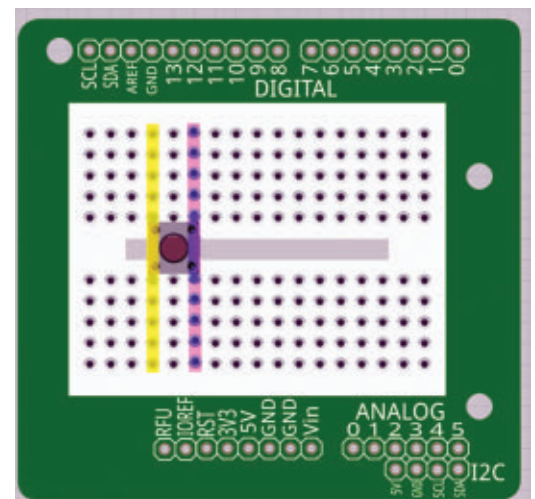


■ **FIGURE 3:** *Pushbutton breadboard and schematic symbols.*

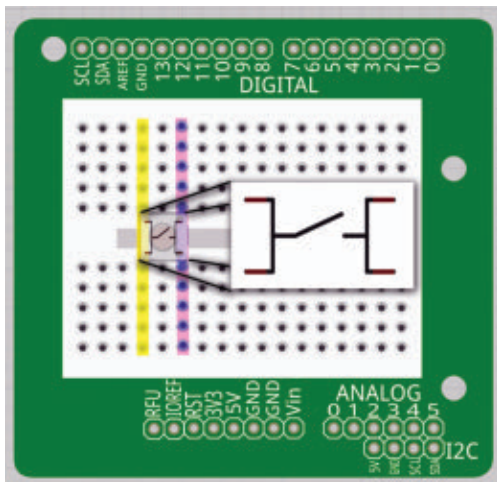
The Arduino Pushbutton

Figures 1 and **2** show how a pushbutton works. Our pushbutton is a little different. Instead of having one connection on each side of the switch, it has two connections on each side as shown in **Figure 3**.

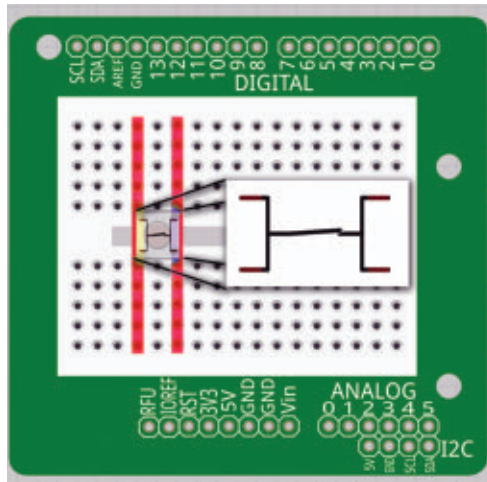
This can be a little confusing. In **Figure 4**, there's a pushbutton on our Arduino proto shield with the connections highlighted in yellow and purple. As you can see, the left side of the pushbutton connects (shown in yellow) both the top and bottom five-pin columns, and the right side shown in purple connects those upper and lower rows.



■ **FIGURE 4:** *Connections when not pushed.*



■ FIGURE 5: Open pushbutton schematic symbol superimposed.



■ FIGURE 6: Closed pushbutton schematic symbol superimposed.

To make this clearer, **Figures 5 and 6** show the pushbutton schematic symbol superimposed on the breadboard symbol with the pushbutton open and then closed. As you can see, when the pushbutton is closed (pushed) you now have both the left and right columns connected (shown in red).

Be sure and remember when you use a pushbutton that the pins come out on only two sides of the pushbutton — the top and bottom (as shown in the illustration) — and that the pins have the switch between them on the side they

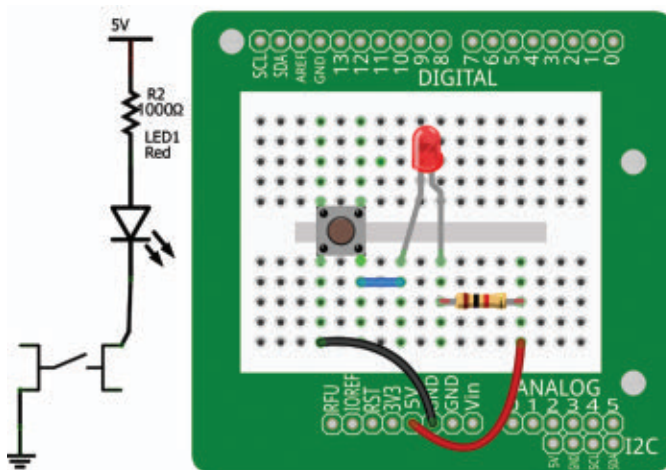
come out of, but are connected to the pin on the other side. If this is still a bit confusing, things should get clearer as we get into the lab exercises.

Lab 1: Pushbutton with LED – Analog.

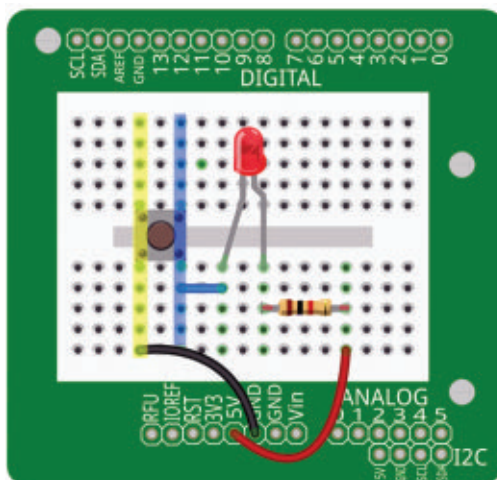
Before we learn to use pushbuttons with the computer (digital), let's first look at how they work electrically (analog). In Chapter 2, we tested an LED with an analog circuit. This time, let's add a pushbutton.

Parts required:

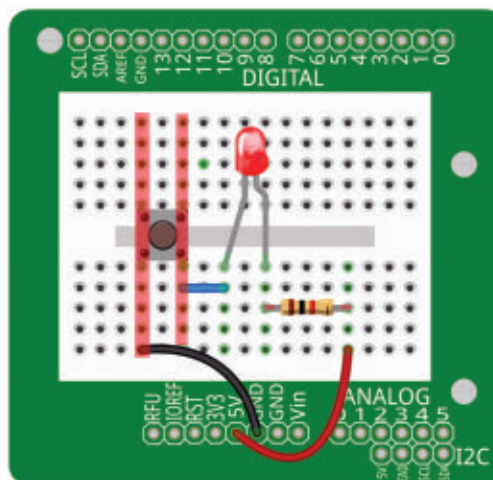
- 1 Pushbutton
- 1 Red LED
- 1 1,000 Ω resistors (brown/black/red)
- 2 Jumper wires



■ FIGURE 7: Pushbutton LED normally off analog circuit.



■ FIGURE 8: Open pushbutton leaves LED off.



■ FIGURE 9: Closed pushbutton turns LED on.

(Notice that **Figures 8 and 9** correspond exactly to **Figures 1 and 2** that show the current flow.)

Turn the LED on with the pushbutton.

Check off when complete:

- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit as shown in **Figure 7**.
- ☐ Apply power to the circuit.

As you can see from **Figure 8**, the pushbutton is open and does not make a

connection, so the circuit is broken and the LED does not light up. When you press the button, the LED will light up.

- ☐ Is the LED on or off?
- ☐ Press the button and see if the LED comes on and stays on while the button is pressed.

Take a moment to think about how the pins on the breadboard are connected by the pushbutton when it is open (unpressed) and closed (pressed). **Figures 8 and 9** show these two states

Turn the LED off with a pushbutton – analog.

Let's rebuild the circuit so that the LED is normally on, and goes off only if the button is pushed. Note from the schematic in **Figure 10** that the current will flow from 5V through the resistor and the LED to ground, but in **Figure 11** the current will flow from 5V through the switch to ground. The reason it flows through the switch and not the LED is that the switch has near zero resistance, while the LED has a relatively higher resistance. Current (like water) flows down the path of least resistance.

Check off when complete:

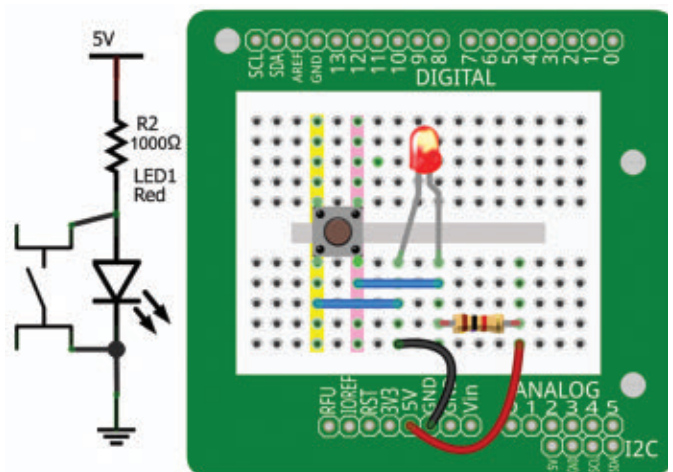
- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit as shown in **Figure 10**.
- ☐ Apply power to the circuit.
- ☐ Is the LED on or off when the button is not pushed?
- ☐ Push the button to verify that the LED turns off while the button remains pushed as shown in **Figure 11**.
- ☐ Briefly explain why the LED is on when the button is not pressed, and why it turns off when the button is pressed.

Lab 2: Reading a Pushbutton – Digital.

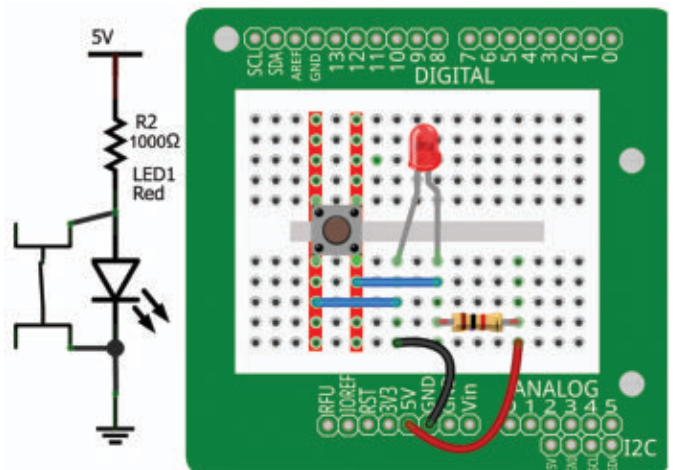
When an Arduino pin is set to the input mode, it may be used to determine if the pin is exposed to a higher or lower voltage. In the design shown in **Figure 12**, pin 12 is connected to a 10 K Ω resistor to ground. The pin is also connected to a pushbutton that is connected to five volts.

When the button is not pushed, the pin is exposed to zero volts and the Arduino software will read this as a LOW. If the button is pushed, then the pin is connected to both the 10 K Ω resistor to ground and to the five volt supply.

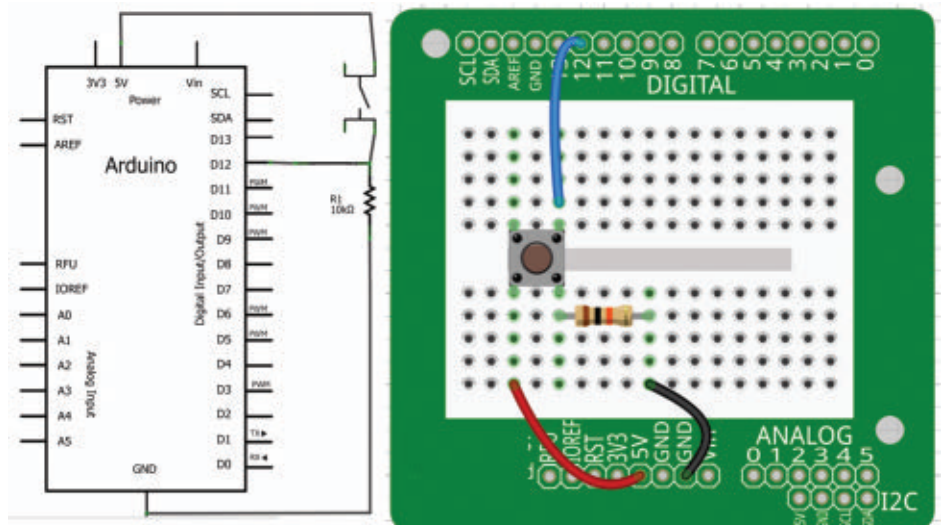
Now, the pin detects that the current is running from the five volts



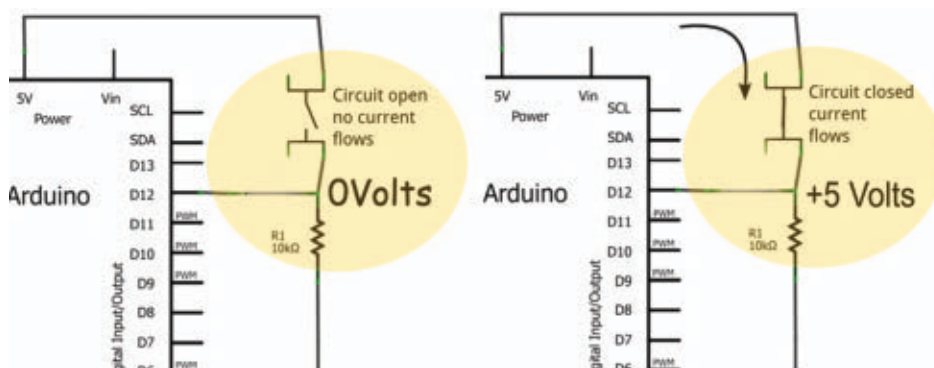
■ FIGURE 10: Pushbutton LED normally on analog circuit.



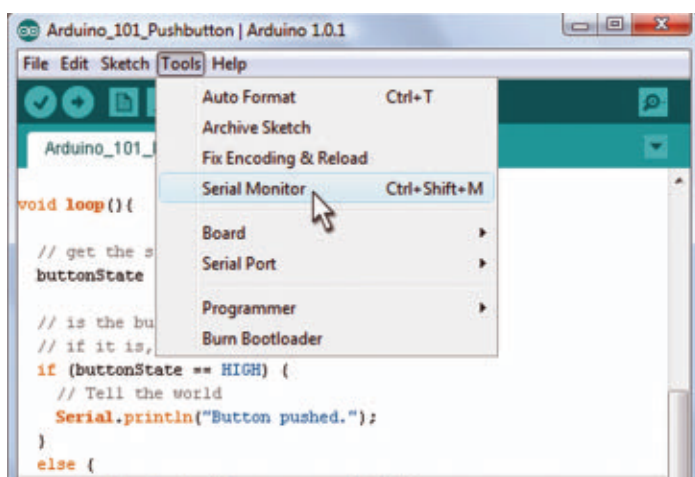
■ FIGURE 11: Pushbutton LED normally on analog circuit – button pushed.



■ FIGURE 12: Pushbutton to digital pin 12.

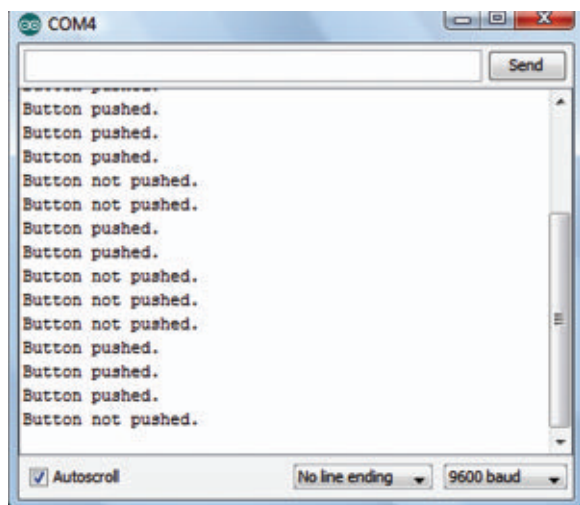


■ FIGURE 13: Pushbutton off and on schematic.



■ FIGURE 14: Open the Serial Monitor.

through the 10 K Ω resistor to ground, so it ‘sees’ the five volts at the positive side of the 10 K Ω resistor. **Figure 13** shows what happens in both cases. Let’s build and test this circuit to help make these concepts clearer.



■ FIGURE 15: Serial Monitor showing button pushes.

Parts required:

- 1 Arduino
- 1 Arduino proto shield
- 1 Pushbutton switch
- 1 10 K Ω resistor (brown/black/orange)
- 2 Jumper wires

Pushbutton Circuit

Check off when complete:

- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit as shown in **Figure 12**.
- ☐ Apply power to the circuit.
- ☐ Open the Arduino IDE and load the *C4_Pushbutton_Serial* program.
- ☐ Verify and upload the program to your Arduino.
- ☐ Open the Tools menu item and click on the Serial Monitor as shown in **Figure 14**.
- ☐ Push the button and release it several times to verify that you get serial output showing the button state as shown in **Figure 15**.

Pushbutton Program

```
// C4_Pushbutton_Serial
// Pushbutton program reports when a button is
// pushed and released

// Constants used to set pin numbers
// (constants can't change while the program
// runs)
const int buttonPin = 12;
// the number of the pushbutton pin

// variables
// (Variables may change while the program runs)
int buttonState = 0;
// variable the pushbutton status

void setup() {
  // initialize Serial communications
  Serial.begin(9600);

  // set the buttonPin mode to INPUT
  pinMode(buttonPin, INPUT);
}

void loop(){

  // get the state of the pushbutton
  buttonState = digitalRead(buttonPin);

  // is the button pressed?
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // Tell the world
    Serial.println("Button pushed.");
  }
  else {
    Serial.println("Button not pushed.");
  }

  delay(500); // pause for 1/2 a second
}
```

Lab 3: Using a Pushbutton for Digital Control of an LED.

In Lab 1, we saw how to use a pushbutton to turn an LED on and off by making or breaking an analog circuit. In this lab, we'll see how to use the Arduino to read the pushbutton state and then decide whether to turn the LED on or off.

Parts required:

- 1 Arduino
- 1 Arduino proto shield
- 1 Pushbutton
- 1 LED
- 1 1 K Ω resistor (brown/black/red)
- 1 10 K Ω resistor (brown/black/orange)
- 5 Jumper wires

Pushbutton LED Circuit

[ASIDE: **Figure 18** shows a photo of a physical realization of the circuit shown in **Figures 16** and **17**. Note that the jumper wires are not the same color as in the diagram, and that they obscure the view of the board. You'll need to be extra careful when building the boards to make sure that you've connected the wires shown in the diagrams and schematics.]

Use a Pushbutton to Turn an LED On

Check off when complete:

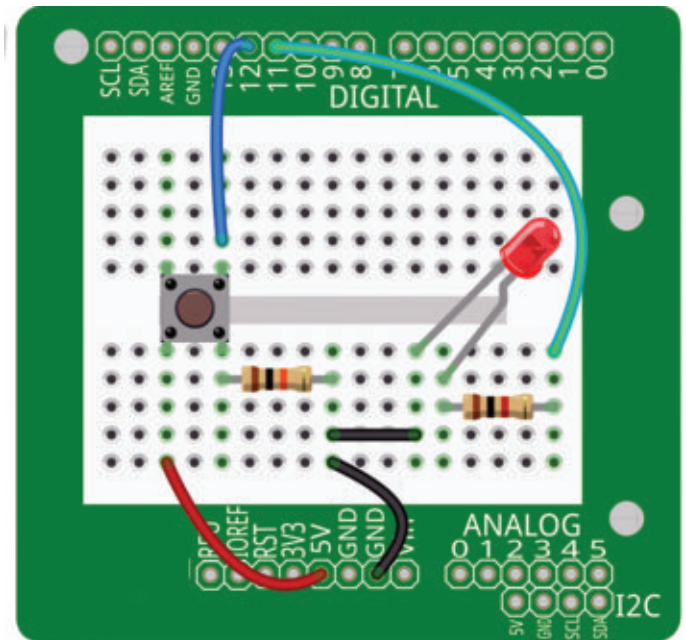
- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit shown in **Figures 16** and **17**.
- ☐ Plug the USB cable into the Arduino.
- ☐ Apply power to the circuit.
- ☐ Open the Arduino IDE and load the `C4_Pushbutton_LED` program.
- ☐ Verify and upload the program to your Arduino.

```
// C4_Pushbutton_LED
// Program turns LED on or off depending on
// pushbutton state
```

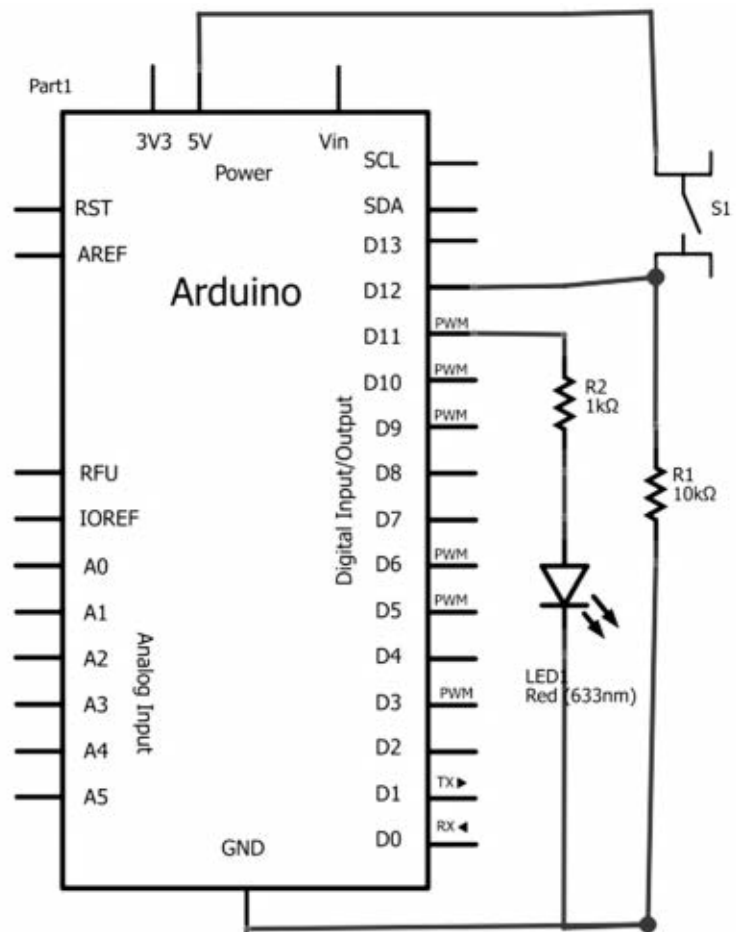
```
// Constants used to set pin numbers
const int buttonPin = 12;
// the number of the pushbutton pin
const int ledPin = 11;
// the number of the LED pin
```

```
// variables
int buttonState = 0;
// variable the pushbutton status
```

```
void setup() {
  // set the buttonPin mode to INPUT
  pinMode(buttonPin, INPUT);
  // set the ledPin mode to OUTPUT
  pinMode(ledPin, OUTPUT);
}
```



■ FIGURE 16: Breadboard pushbutton LED digital control.



■ FIGURE 17: Schematic pushbutton LED digital control.

```

}

void loop() {

    // get the state of the pushbutton
    buttonState = digitalRead(buttonPin);

    // is the button pressed?
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }
    else {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }

    delay(500); // pause for 1/2 a second
}

```

Use a Pushbutton to Turn an LED Off

We can easily reverse the way this system works so that instead of turning the LED on when the button is pressed, it can turn the LED off — simply by changing only the two lines highlighted in the `loop()` function shown here.

[The comments are also changed to match the functional change, but only the source needs to be changed to reverse the way the button works.]

My friend Jay Flanders pointed out that this is the

beauty of programmable devices — you can change the software instead of the hardware.

Check off when complete:

- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit shown in **Figures 16 and 17**.
- ☐ Plug the USB cable into the Arduino.
- ☐ Apply power to the circuit.
- ☐ Change the `C4_Pushbutton_LED` program to the two highlighted lines shown in the `loop()` function below.
- ☐ Verify and upload the program to your Arduino.

```

void loop() {

    // get the state of the pushbutton
    buttonState = digitalRead(buttonPin);

    // is the button pressed?
    // if it is, the buttonState is HIGH:
    if (buttonState == HIGH) {
        // turn LED off:
        digitalWrite(ledPin, LOW);
    }
    else {
        // turn LED on:
        digitalWrite(ledPin, HIGH);
    }

    delay(500); // pause for 1/2 a second
}

```

Notice that the only difference in the code is that we reversed the HIGH and LOW in the `digitalWrite()` function, which is shown in the source by highlighting the lines to change.

Use a Pushbutton to Control a Pulsing LED

In the last example, the LED stays on or off only while the pushbutton is pressed or released. This might be inconvenient if we have an LED that is normally off and only comes on when we hold down the pushbutton. What if somebody knocks on the door? We release the pushbutton and the LED goes off while we are answering that door. What we probably really want is a design where we can just press a button to turn the LED on and then come back later and press the button to turn it off.

Let's write a program so that if the LED is off and the button is pressed and released, the LED will come on. Then, if the LED is on and the button is pressed and released, the LED will turn off.

Think about how you might do

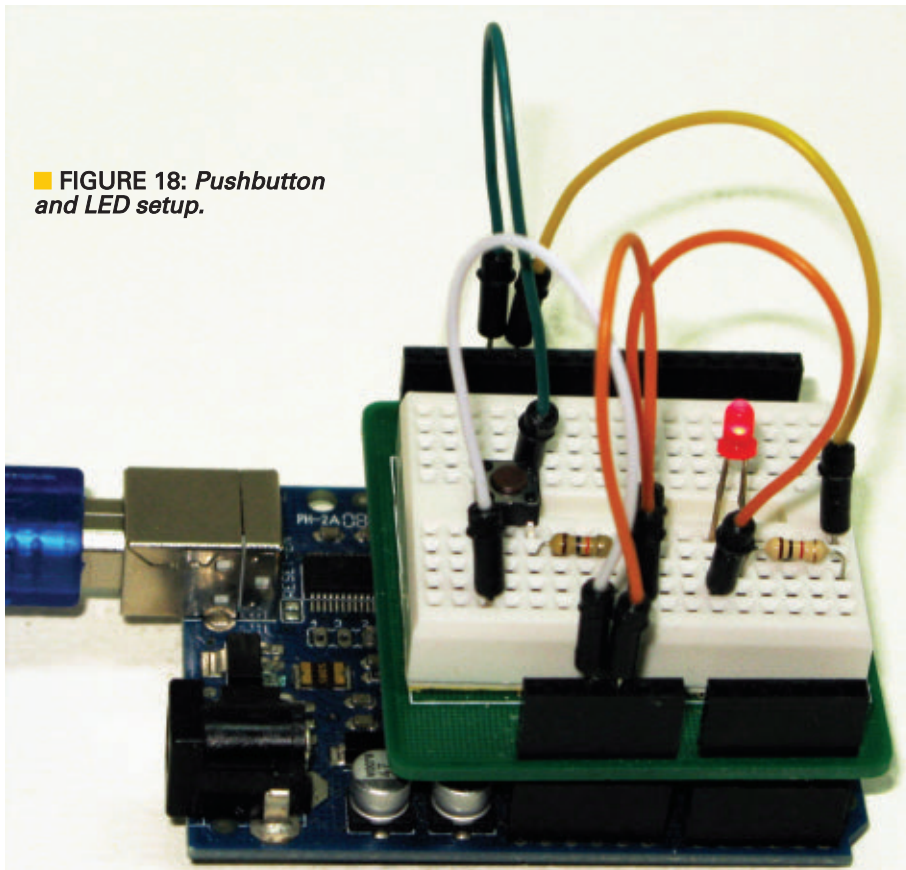


FIGURE 18: Pushbutton and LED setup.

this. The microcontroller will be running through the *loop()* function, and each time through it will need to decide to turn the LED on or off as with the earlier programs; in them, however, it checked the state of the pushbutton to make the on/off decision.

What we want is a variable that the *loop()* can examine to see if the LED should be on or off. For instance, we may create a global variable *ledState* and set it to zero. Then, when we are in the *loop()* function, we will check the pushbutton. If it is pressed, we will check the *ledState*; if that state is zero we will change it to one, and if it is one we will change it to zero. This action is called **toggle**; we are said to **toggle** the state.

We will also use a *delay* so we can get our finger off the pushbutton before the *loop()* goes around and checks it again.

Check off when complete:

- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit shown in **Figures 16 and 17**.
- ☐ Plug the USB cable into the Arduino.
- ☐ Apply power to the circuit.
- ☐ Open the Arduino IDE and load the *C4_Pushbutton_LED_state_change* program.
- ☐ Verify and upload the program to your Arduino.

```
// C4_Pushbutton_LED_state_change
// Changes the LED state each time the
// pushbutton is pressed.

// Constants used to set pin numbers
const int buttonPin = 12;
// the number of the pushbutton pin
const int ledPin = 11;
// the number of the LED pin

// variables
int buttonState = 0;
// variable the pushbutton status
int ledState = 0;
// variable the LED 0 is off, 1 is on

void setup() {
  // set the buttonPin mode to INPUT
  pinMode(buttonPin, INPUT);
  // set the ledPin mode to OUTPUT
  pinMode(ledPin, OUTPUT);
}

void loop(){

  // get the state of the pushbutton
  buttonState = digitalRead(buttonPin);

  // set the LED state based on the pushbutton
  // state
  if(buttonState)
  {
    if(ledState)
    {
      ledState = 0;
    }
    else ledState = 1;
  }
  delay(500); // Give time to release the button
```

```
// is the button pressed?
// if it is, the buttonState is HIGH:
if (ledState == 1) {
  // turn LED on:
  digitalWrite(ledPin, HIGH);
}
else {
  // turn LED off:
  digitalWrite(ledPin, LOW);
}
}
```

Lab 4: Testing Your Reaction Time.

Parts required:

- 1 Arduino
- 1 Arduino proto shield

Check off when complete:

- ☐ Make sure the power is off before building the circuit.
- ☐ Build the circuit shown in **Figures 16 and 17**.
- ☐ Plug the USB cable into the Arduino.
- ☐ Apply power to the circuit.
- ☐ Open the Arduino IDE and load the *C4_Time_Tester* program.

```
// C4_Reaction_Time_Tester
// Reports over the serial port how many milli
// seconds it takes you to remove your finger
// from the pushbutton after the LED turns on.

// Constants used to set pin numbers
const int buttonPin = 12;
// the number of the pushbutton pin

// variables
int buttonState = 0;
// pushbutton status variable
const int ledPin = 11;
// the number of the LED pin
int reactionTime = 0;
// reaction time variable

unsigned long startTime = 0; //
unsigned long endTime = 0; //

long randNumber;

void setup() {
  // initialize Serial communications
  Serial.begin(57600);
  Serial.println("Test your reaction time.");
  Serial.println("Hold down button and release
  when LED comes on.");

  // set the buttonPin mode to INPUT
  pinMode(buttonPin, INPUT);
  // set the ledPin mode to OUTPUT
  pinMode(ledPin, OUTPUT);
}

void loop(){

  // get the state of the pushbutton
  buttonState = digitalRead(buttonPin);
```

Imagine this...

- a small and inexpensive
- USB or ethernet device
- with a rich set of features
- including PLC functionality and 3-axis stepper motor controller
- all accessible via free .NET, ActiveX or C++ library
- cross-platform
- configurable with free software

PoKeys



Or this...

- Smallest USB 2.0 portable 1MS/s oscilloscope
- Data acquisition of analog and digital signals
- Data recording
- Export to CSV, XLS, PDF and HTML
- Simple usage of advanced features
- Examples for C++, VB, Delphi and LabView
- Free software and updates

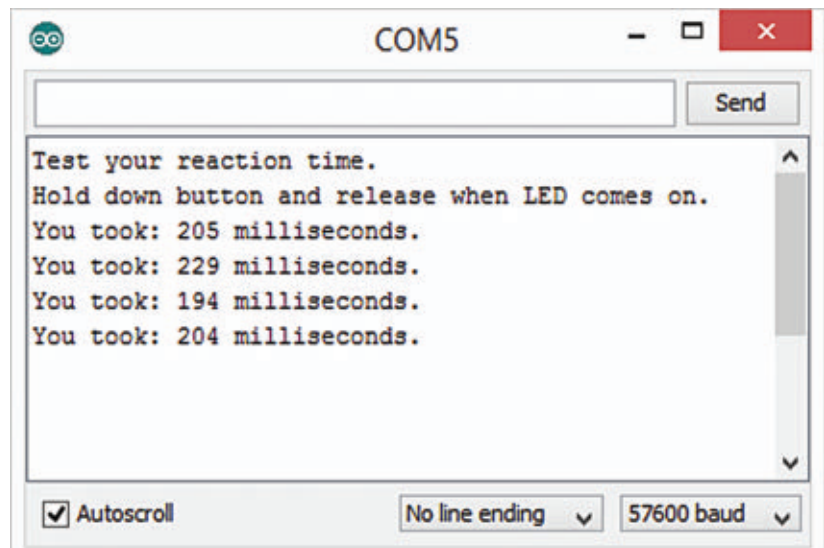
PoScope Mega1+



PoLabs

Visit us at

www.poscope.com



■ FIGURE 19: Reaction time.

```
// is the button pressed?
// if it is, the buttonState
// is HIGH:
if (buttonState == HIGH) {

    // select a random number
    // of milliseconds
    // from 1000 to 5000
    randomNumber = random
    (1000,5000);

    // turn LED on:
    digitalWrite(ledPin, HIGH);

    // leave the LED on for
    // the random
    // milliseconds
    delay(randomNumber);

    // turn LED off:
    digitalWrite(ledPin, LOW);

    // get the start time as
    // soon as the LED
    // goes off
    startTime = millis();

    // wait until the subjects
    // gets his finger
    // off the button
    // read the button state
    // until it is equal
    // to HIGH
    do{
        // get the state of the
        // pushbutton
        buttonState = digital
        Read(buttonPin);
    }while(buttonState ==
    HIGH);

    // get the end time as
    // soon as the finger
    // is off the switch
    endTime = millis();
```

```
// Tell the world your
// reaction time
Serial.print("You took: ");
Serial.print(endTime-start
Time,DEC);
Serial.println(" milli
seconds.");
}
```

- ☐ Verify and upload the program to your Arduino.
- ☐ Open the Arduino serial monitor.
- ☐ Press and hold the pushbutton.
- ☐ When the LED goes off, release the button.
- ☐ Verify that your reaction time is being shown in the serial monitor as shown in **Figure 19**.
- ☐ Do this 10 or more times, and then average your reaction time.
- ☐ Challenge someone to a contest and see who has the fastest reaction time.

That's it for this month. Now, go get pushy. **NV**

ELECTRONET

GET THE NUTS&VOLTS DISCOUNT!
Mention or enter coupon code **NVRMZ142**
and receive 10% off your order!

ramsey
www.ramseykits.com
AM/FM Broadcasters • Hobby Kits
Learning Kits Test Equipment
...AND LOTS OF NEAT STUFF!

CORIDIUM
ARM controllers running
floating point **BASIC** from \$10.00
www.coridium.us

Got electronics?
www.nutsvolts.com

FREE PCB Design Software
Get Started Now
at **FabStream.com**
FabStream
A Downstream Technologies Solution

INVEST in your BOT!

HiTEC
12115 Paine Street • Poway, CA 92064 • 858-748-6948 • www.hitecusa.com

HOBBY ENGINEERING
Kits, Parts and Supplies
www.HobbyEngineering.com

MyRO www.myropcb.com
PCB, PCBA and More
LOW cost with HIGH Quality
1-888-PCB-MYRO

ALL ELECTRONICS CORPORATION
Electronic Parts and Supplies.
www.allelectronics.com
Free 96 page catalog 1-800-826-5432

For the ElectroNet online,
go to
www.nutsvolts.com
click **Electro-Net**

THE ORIGINAL SINCE 1994
PCB-POOL
Beta LAYOUT
• Low Cost PCB prototypes
• Free laser SMT stencil
with all Proto orders
WWW.PCB-POOL.COM

MOBILE APP NOW AVAILABLE!

SERVO
Download NOW On Your
Favorite Mobile Device!

FOR ROBOT BUILDERS
iOS • ANDROID • KINDLE FIRE

USB Add USB to your next project--
It's easier than you might think! **DLP Design**
USB-FIFO • USB-UART • USB/Microcontroller Boards
RFID Readers • Design/Manufacturing Services Available
Absolutely NO driver software development required!
www.dlpdesign.com

superbrightleds.com
COMPONENT LEDs • LED BULBS • LED ACCENT LIGHTS

LED modules
Solar controls
Dusk to dawn
Dimmers 12/24V
J2 LED LIGHTING LLC
www.j2ledlighting.com

NATIONAL RF, INC.
www.NationalRF.com
RF, DF, WxSat/Ham Gear, Dials
Attenuators, Plug-in Coils

Total Eclipse of the Cross Compiler

Post comments on this article and find any associated files and/or downloads at www.nutsvolts.com/index.php?/magazine/article/april2014_DesignCycle.

If you've ever programmed a PIC or AVR microcontroller, you most likely built your embedded application using cross compilation techniques. A cross compiler environment running on your PC is necessary in this case because the PIC and AVR embedded microcontrollers do not have the memory and CPU resources that are necessary to handle a kernel-based operating system and all of the supporting firmware. You won't get utilities, libraries, and compilers to fit in those small memory spaces either. The BeagleBone Black and Raspberry Pi are complete embedded systems that happen to have the capacity to absorb kernels, libraries, utilities, applications, and compilers. Thus, it is possible to build and deploy an application using nothing but a Pi or BeagleBone and the Linux or Android operating system the board is running. The bottleneck is speed. My Pi and BeagleBone can't perform compilations as quickly as my Lenovo ThinkPad. So, it is advantageous to do my compiling on the ThinkPad and download the resultant binary to the target Pi or Black microcontroller.

Nothing Beats Free

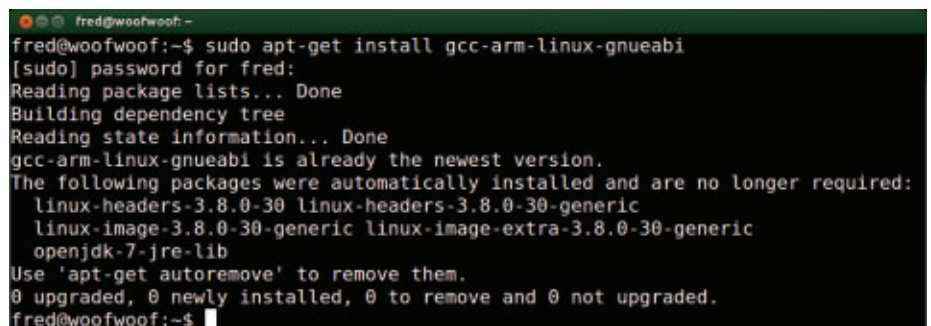
One of the big draws to the Raspberry Pi and BeagleBone Black is their ability to run free open source software and operating systems. The BeagleBone and Pi are relatively cheap, and it costs nothing but your time to program them. The same holds true on the PC side of the compilation equation.

Linux is available as a free download and so are the cross compilation tools you will need to feed you dog and bake a Pi. Both the Pi and BeagleBone can utilize the ARM versions of the GCC (GNU Compiler Collection). The GNU compilers, linkers, and assemblers have the ability to fall under the control of another free open source development tool called Eclipse.

We can load the GNU compilers, linkers, utilities, and assemblers directly onto our Pi and BeagleBone. However, the GNU compilers and development tools are better utilized when loaded as cross compilation systems on a more powerful PC. I have chosen to run the Pi and BeagleBone GNU C and C++ cross compilers on an old ThinkPad T61 running Ubuntu 13.04.

It is relatively easy to load the GNU C/C++ compilers on my Linux laptop. One must simply open a Ubuntu terminal window and issue `sudo apt-get install g++-arm-linux-gnueabi`. As you can see in **Screenshot 1**, my Linux laptop is up to date. Otherwise, the ARM GNU cross compiler would be downloaded and installed.

This particular ARM cross



```
fred@woofwoof:~$ sudo apt-get install gcc-arm-linux-gnueabi
[sudo] password for fred:
Reading package lists... Done
Building dependency tree
Reading state information... Done
gcc-arm-linux-gnueabi is already the newest version.
The following packages were automatically installed and are no longer required:
  linux-headers-3.8.0-30 linux-headers-3.8.0-30-generic
  linux-image-3.8.0-30-generic linux-image-extra-3.8.0-30-generic
  openjdk-7-jre-lib
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
fred@woofwoof:~$
```

■ Screenshot 1. Linux is Linux is Linux. If you can get your BeagleBone Black on the Internet, you can issue this same command and get the same results that a PC running Linux would produce.



■ Screenshot 3. I fiddled with every entry except the Source. Most of the fields will end up as description comments.

compiler targets the BeagleBone. To get the Pi cross compiler tools from a Linux terminal, you would issue *git clone git://github.com/raspberrypi/tools.git*. This will clone the tools into your selected directory. You can also use a Linux terminal and *https* to gain access to the Pi cross compiler tools using *https://github.com/raspberrypi/tools.git*.

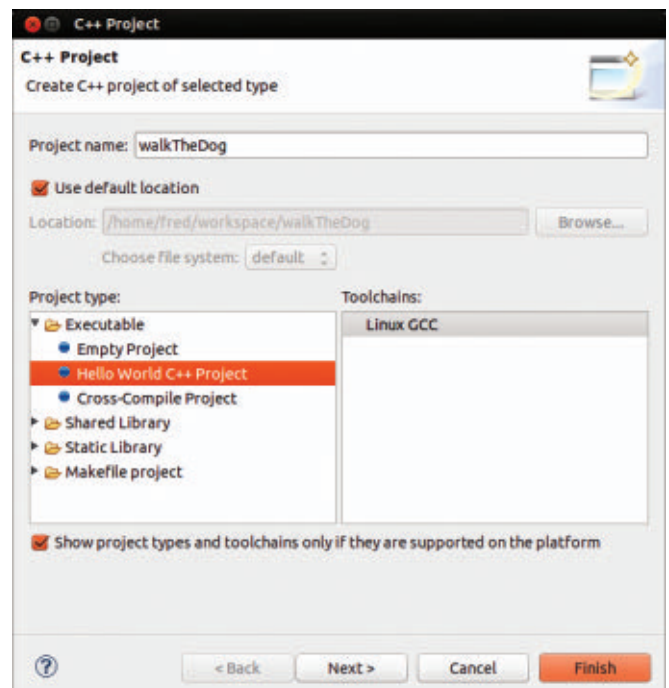
I Hate Hello World Programs

So, our very first cross compiled C++ BeagleBone program will fall under the project name of *walkTheDog*. Instead of “Hello World!” our program will display “Design Cycle Walks the Dog.” Let’s put Eclipse to work beginning with **Screenshot 2**.

The Hello World C++ Project selection will force us to use the Linux GCC toolchain. This selection will also write a simple C++ skeleton Hello World-like program for us. We can control some of the content of the pregenerated C++ program that Eclipse will produce by filling in the blanks in **Screenshot 3**. Most of the text will end up behind comment characters in the main file. The greeting field will be transferred to the main function call.

The results of my interaction in **Screenshot 3** can be seen in **Screenshot 4**. The *iostream* declaration enables the use of the standard output object *cout*. Declaring *using namespace std*; eliminates the need to explicitly define the standard namespace for *cout* using a namespace prefix (*std::cout*) in the code.

If we were to build the C++ code right now, it



■ Screenshot 2. Our anti-Hello World code begins here. Note the Linux GCC toolchain has been assigned. Don’t be fooled as this is not yet explicitly pointing at the ARM GCC toolchain. Note the project location (/home/fred/workspace/walkTheDog).

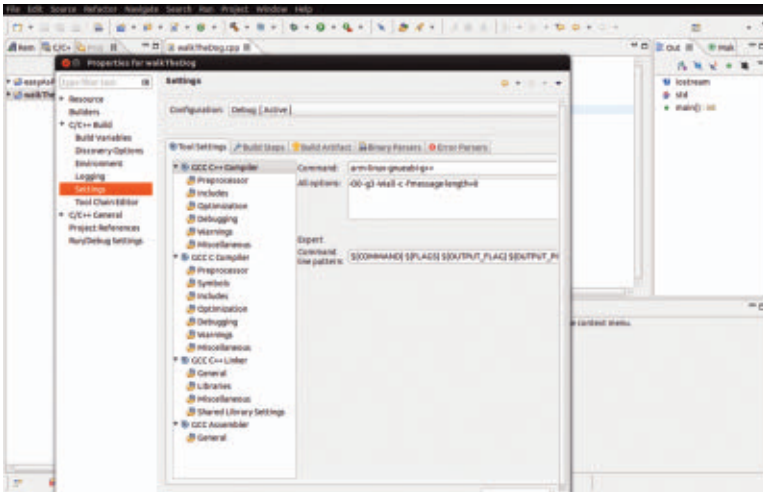
would not run on our BeagleBone. That’s because the Linux GCC toolchain that is being referenced belongs to the ThinkPad’s Ubuntu distribution and would produce Ubuntu i386 binaries instead of ARM binaries. To cross compile, we’ve got to point to and use the ARM *include* directories and libraries. We must also make sure we use the correct ARM compilation commands.

ARMing the Compilers

The default command for the GCC C++ compiler in **Screenshot 5** was *g++*. As you can see, I’ve changed it to



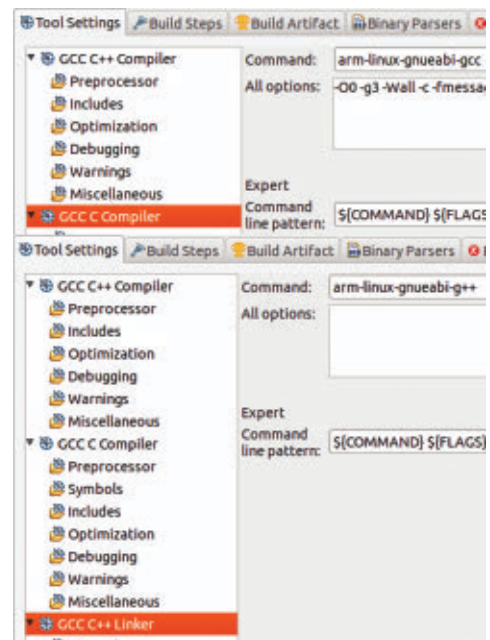
■ Screenshot 4. Everything I changed in Screenshot 3 can be seen here. This is just about as basic as a C++ program can get.



■ Screenshot 5. There are lots of knobs to twist and buttons to push here. However, right now we only have to worry about the compiler, linker, and assembler ARM command strings.

read *arm-linux-gnueabi-g++*. We will also have to manually alter the commands for the GCC C compiler, the GCC C++ linker, and the GCC assembler as well. I've done just that in **Screenshot 6**.

Now that the commands are ARMed, the next step is to make sure the correct ARM *includes* and ARM libraries will be accessed during the cross compilation and linking processes. Note that PC *include* paths are present in the project properties window I've captured in **Screenshot 7**. Our job is to add an *include* directory path for the BeagleBone. The path */usr/arm-linux-gnueabi/include* will be added at the top of the list. We will do the same for the ARM C++ compiler in **Screenshot 8**. Once we have ARMed the *includes*, we do the same for the ARM

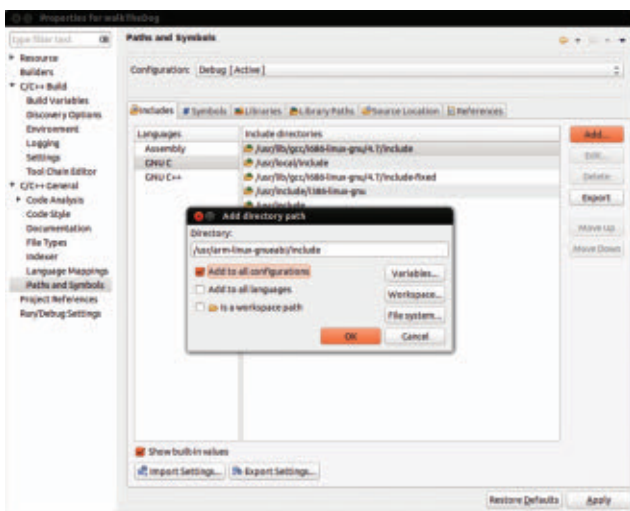


■ Screenshot 6. In the end, all we must do is add the prefix *arm-linux-gnueabi-* to all of the compilers, linkers, and assemblers that are represented in this screen capture. The assembler command is missing here, however. It reads as *arm-linux-gnueabi-as*.

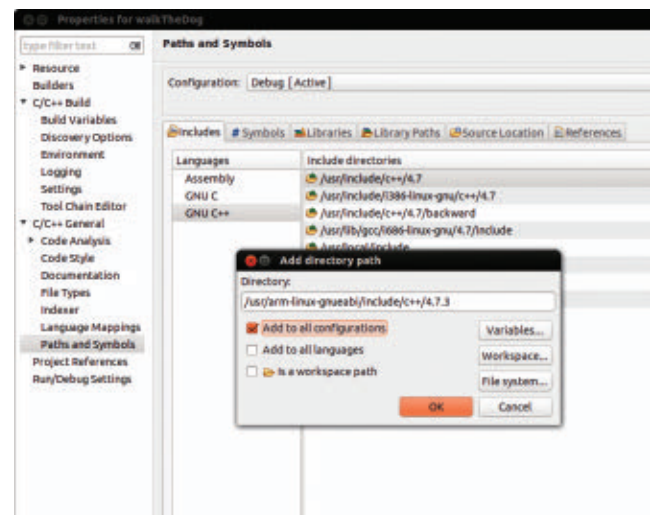
libraries. **Screenshot 9** holds the directory path entry for the ARM libraries.

Pulling the Trigger

If we have done everything correctly, clicking on the



■ Screenshot 7. The BeagleBone Black *includes* are located in the path */usr/arm-linux-gnueabi/include*. This directory path happens to exist on the ThinkPad.



■ Screenshot 8. To use both of the C and C++ ARM cross compilers, we must perform the same directory pathing process for each compiler we wish to use. In the case of C++, the *include* path is */usr/arm-linux-gnueabi/include/c++/4.7.3*.

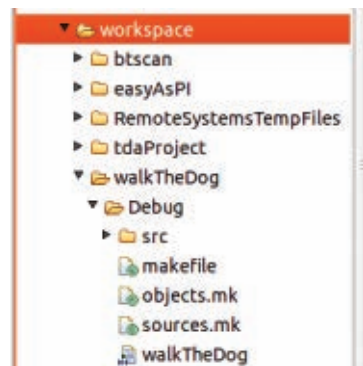
Eclipse Build icon will compile and link our C++ source code. **Screenshot 10** is the console log of the *walkTheDog* project's build process. Although you can't see all of the window contents in **Screenshot 5**, you can still put together some of what you see in **Screenshot 10** to what you see in **Screenshot 5**.

You can see that the GCC C++ compiler invocation string we entered in **Screenshot 6** is used to kick off the compiler (*arm-linux-gnueabi-g++*). The cross compile options including the C++ *include* path we specified in **Screenshot 8** can also be identified in the console screen capture. Once the source file *walkTheDog* is built, the GCC C++ linker is invoked using *arm-linux-gnueabi-g++*. You can also see that the library path we added in **Screenshot 9** is referenced in the ARM linker command (*/usr/arm-linux-gnueabi/lib*).

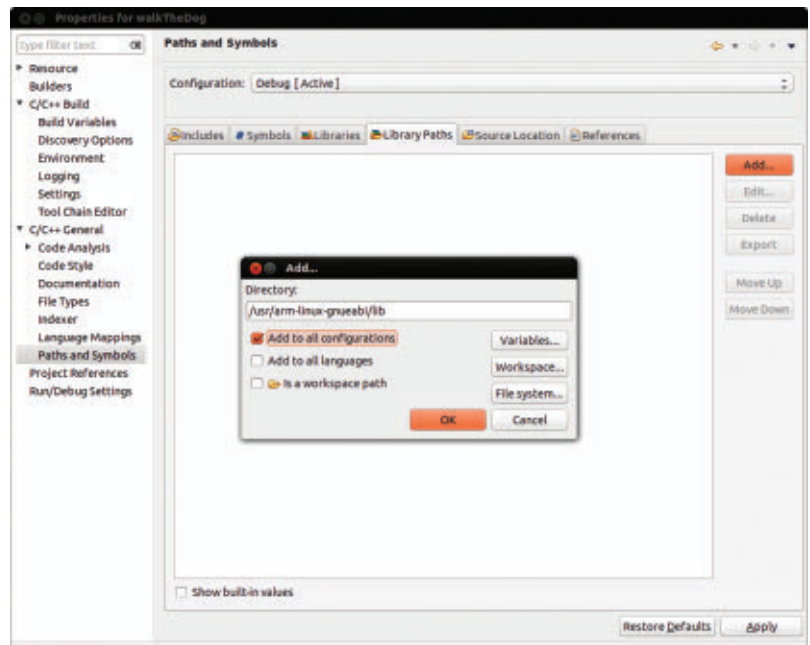
The compilation and linking process completed successfully and generated a binary file called *walkTheDog*. To the left of the capture, you can see the newly created *walkTheDog* binary file has been placed in the default workspace directory. The actual path to *walkTheDog* is */home/fred/workspace/walkTheDog*. This path was established in the C++ project window captured in **Screenshot 2**. A better view of the ThinkPad local directory structure exists in **Screenshot 11**.

Walking the Dog

We now have a binary file in our ThinkPad *workspace* directory that was built to run on the



■ Screenshot 11. This is a shot of the ThinkPad local workspace directory structure. The Eclipse IDE allows us to drag and drop files between the ThinkPad workspace directory and the BeagleBone Black file system.

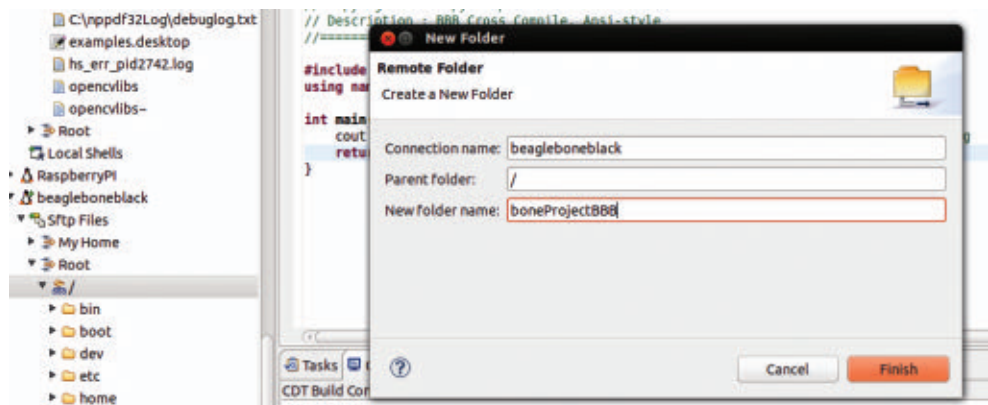


■ Screenshot 9. The ARMin procedure for the BeagleBone Black libraries is identical to the process used to ARM the *includes*.

BeagleBone. So, we have to move the binary file from the ThinkPad *workspace* directory to the BeagleBone's



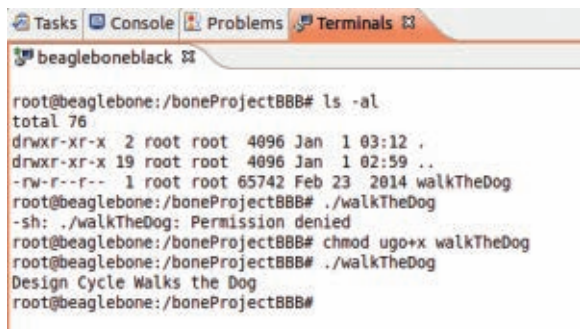
■ Screenshot 10. The compilation and linking steps captured here follow the command structures, *include* pathing, and library pathing we established in Screenshot 5.



■ Screenshot 12. I was able to create a new directory on the BeagleBone Black from the Eclipse cross compilation session running on the ThinkPad using the file and directory resources of Eclipse.

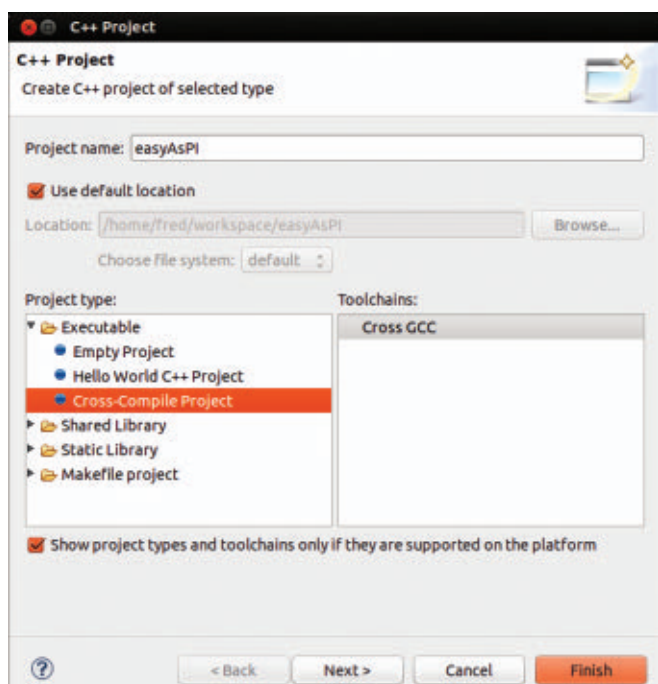


■ Screenshot 13. It's Linux but it works just like Windows. After I created the *boneProjectBBB* folder, I simply dragged the *walkTheDog* file from the ThinkPad local workspace folder to the BeagleBone *boneProjectBBB* folder.



■ Screenshot 14. Thus far, we have done most everything through the Eclipse session running on the ThinkPad. It should be no surprise that we can easily interact with the BeagleBone from it.

Angstrom file system. Instead of just dragging the *walkTheDog* ARM file to the BeagleBone root directory, let's use Eclipse to create a folder on the BeagleBone and



■ Screenshot 15. Note that the Linux GCC toolchain we used with the BeagleBone Black is not supported by the Raspberry Pi's Cross-Compile Project configuration.

copy the file from the ThinkPad to the newly created folder.

In **Screenshot 12**, I've navigated to the BeagleBone file system from within the Eclipse cross compilation session and created a folder called *boneProjectBBB* in the BeagleBone's root directory.

To copy the *walkTheDog* from the ThinkPad to the BeagleBone, I dragged it from the ThinkPad folder and dropped it into the BeagleBone folder as shown in **Screenshot 13**.

Screenshot 14 is a capture of the BeagleBone terminal session I kicked off from the ThinkPad's Eclipse cross compilation session. As you can see, everything didn't go as planned. Following the listing of the contents of the *boneProjectBBB*, I attempted to run the *walkTheDog* application with no joy.

A closer look at the file permissions (and error message) shows that the *walkTheDog* file cannot be executed by any user. Not to worry. All we have to do is activate execution privileges for the binary file via the *chmod* command. The results are obvious to the most casual observer.

KP Duty

Baking things in the kitchen isn't much different than taking the dog for a walk. The most notable difference is that the Pi requires a different version/flavor of the GNU C++ compiler. It also differs in the way we assemble the Eclipse project environment. As you can see in **Screenshot 15**, for the Pi we have chosen the Cross-Compile Project instead of the Hello World C++ Project. The Linux GCC toolchain components we used to support the BeagleBone are not supported by the Cross-Compile Project configuration.

Setting up the Pi Eclipse environment was a bit less work than the BeagleBone implementation. Instead of entering the command prefix for each compiler, linker, and assembler, the command prefix for the Pi only needed to be entered once.

The path entry you see in **Screenshot 16** worked like magic. Once entered, I proceeded to the paths window to enter the *include* and library paths. To my surprise, the entries were already populated. I captured the C++ *include* directory paths in **Screenshot 17**.

At this point, everything we've done to cross compile with the BeagleBone is done again with the Pi. Our Pi source file — which is displayed in **Screenshot 18** — looks a lot like our BeagleBone source file. The Pi remote session area of the Eclipse session can be seen in **Screenshot 19**. The file movement and folder creation

The Raspberry and BeagleBone can be had from a number of distributors. You can source them from your preferences of distributors. Here's a few I know of:

Raspberry Pi:

Newark
AdaFruit
Newegg
Allied
MCM

Beaglebone Black:

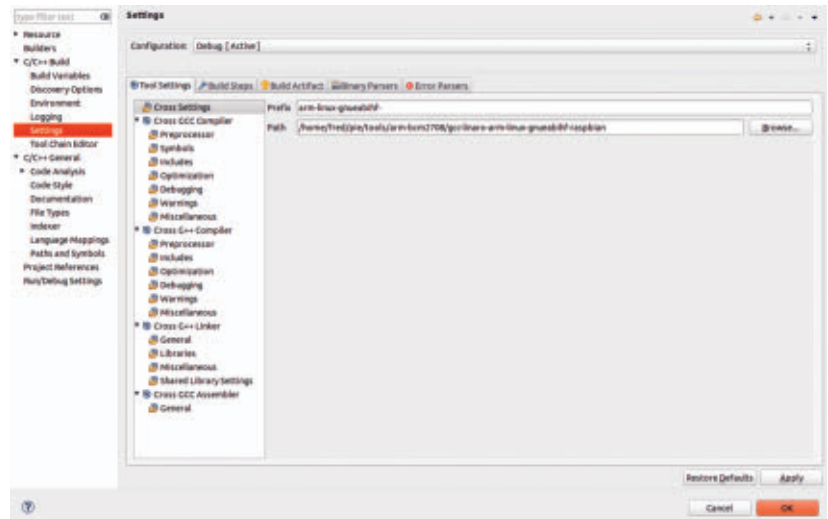
Newark
Digi-Key
Mouser

tasks that worked with the BeagleBone also work in the exact same way with the Pi.

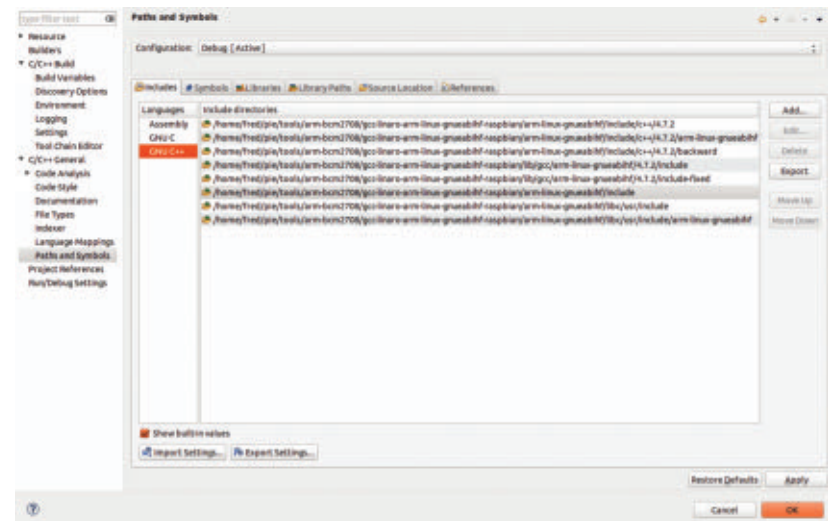
Once the Pi cross compilation and linking process completed, I was able to drag and drop the *easyAsPI* binary file into the Pi's *sliceofPI* folder. In an identical manner, I attempted to execute *easyAsPI* binary without the execute permissions. As with the BeagleBone, the file would not execute until I issued the *chmod* command to enable execute permissions.

Standing at the Cross Roads

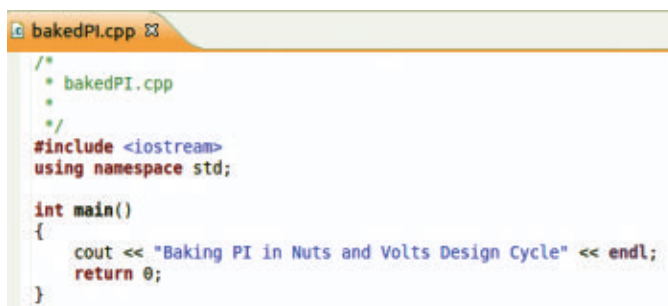
This discussion began by introducing the BeagleBone and Pi in the context of software development. Last time, we loaded and updated the BeagleBone and Pi Linux operating systems. This time around, we have built a Linux cross compilation development system on a discarded ThinkPad laptop, and cross compiled simple C++ applications for both the BeagleBone and Pi. You now possess the knowledge and tools necessary to build more complex BeagleBone and Pi applications. The Raspberry Pi, BeagleBone Black, Eclipse, GCC, and Linux are now part of your design cycle. **NV**



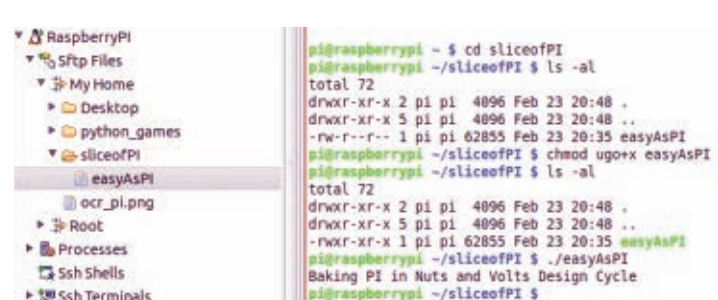
■ Screenshot 16. The Raspbian flavor of the ARM GNU C compiler uses the on-chip hardware floating point unit instead of emulating floating point functions in software. Thus, the *hf* appended to the prefix.



■ Screenshot 17. This was a pleasant surprise. The *include* directory paths were populated based on the compiler main path we specified in Screenshot 16.



■ Screenshot 18. C++ is C++ is C++. Only the displayed message has changed here.



■ Screenshot 19. Eclipse presents a common look and feel for working with the BeagleBone Black and Raspberry Pi.

The Nuts & Volts **WEBSTORE**

GREAT FOR DIYers!

The Steampunk Adventurer's Guide by Thomas Willeford

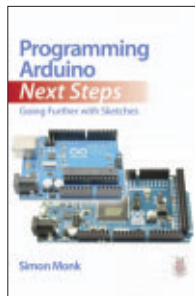
Steampunk stalwart Thomas Willeford cordially invites you on an adventure — one in which you get to build ingenious devices of your own! Lavishly illustrated by award-winning cartoonist Phil Foglio, *The Steampunk Adventurer's Guide: Contraptions, Creations, and Curiosities Anyone Can Make* presents 10 intriguing projects ideal for makers of all ages and skill levels, woven into an epic tale of mystery and pursuit.



\$25.00

Programming Arduino Next Steps: Going Further with Sketches by Simon Monk

Arduino guru Simon Monk reveals advanced programming techniques for Arduino! *Programming Arduino Next Steps: Going Further with Sketches* is the must-have follow-up to Monk's bestseller, *Programming Arduino: Getting Started with Sketches*. Aimed at experienced programmers and hobbyists who have mastered the basics, this book takes you "under the hood" of the Arduino, revealing professional-level programming secrets.



\$20.00

Programming the BeagleBone Black: Getting Started with JavaScript and BoneScript by Simon Monk

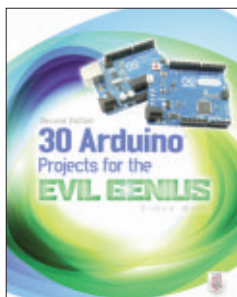
Learn how to program the BeagleBone Black — the wildly popular single-board computer — using JavaScript and the native BoneScript language. You'll find out how to interface with expansion capes to add capabilities to the basic board, and how to create a Web interface for BBB. Two hardware projects demonstrate how to use the board as an embedded platform.



Price \$15.00

30 Arduino Projects for the Evil Genius: Second Edition by Simon Monk

Fully updated throughout, this do-it-yourself guide shows you how to program and build fascinating projects with the Arduino Uno and Leonardo boards, and the Arduino 1.0 development environment. *30 Arduino Projects for the Evil Genius, Second Edition*, gets you started right away with the simplified C programming you need to know, and demonstrates how to take advantage of the latest Arduino capabilities.



\$25.00

Build Your Own Transistor Radios by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits

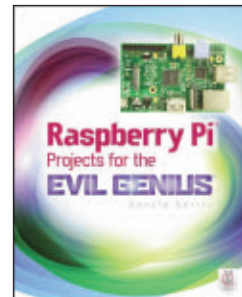
Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.



***Paperback, 496 pages
\$49.95**

Raspberry Pi Projects for the Evil Genius by Donald Norris

This wickedly inventive guide shows you how to create all kinds of entertaining and practical projects with the Raspberry Pi operating system and programming environment. Each fun, inexpensive Evil Genius project includes a detailed list of materials, sources for parts, schematics, and lots of clear, well-illustrated instructions for easy assembly. The larger workbook-style layout makes following the step-by-step instructions a breeze.



\$25.00

How to Diagnose and Fix Everything Electronic by Michael Jay Geier

Master the Art of Electronics Repair

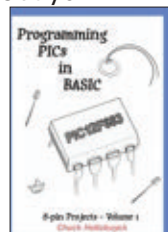
In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.



\$24.95

Programming PICs in Basic by Chuck Hellebuyck

If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller.



\$14.95

Master and Command C for PIC MCUs by Fred Eady

Master and Command C for PIC MCU, Volume 1 aims to help readers get the most out of the Custom Computer Services C compiler for PIC microcontrollers.

The author describes some basic compiler operations that will help programmers particularly those new to the craft create solid code that lends itself to easy debugging and testing. As Eady notes in his preface, a single built-in CCS compiler call (output_bit) can serve as a basic aid to let programmers know about the "health" of their PIC code.



\$14.95

Order online @ www.store.nutsvolts.com
Or CALL 1-800-783-4624 today!

BOOK & KIT COMBOS

Raspberry Three Book Combo



Only \$48.95
Plus
FREE Priority Mail Shipping
US Only

Beginner's Guide Complete Combo!



Combo Price \$229.95

For complete details, visit our webstore @ www.nutsvolts.com.

Arduino Classroom
Arduino 101 Projects Kit



\$44.99

From Smiley's Workshop

CD-ROM SPECIAL

Nuts & Volts
10 CD-ROMs
& Hat Special!

That's 120 issues.
Complete with supporting
code and media files.



Free Shipping!

Only \$208.95
or \$24.95 each.

The Nuts & Volts
Pocket Ref

All the info you need at your fingertips!

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions. AND it still fits in your shirt pocket!

Only \$12.95

Visit <http://store.nutsvolts.com> or call (800) 783-4624

PROJECTS

Super Detector Circuit Set



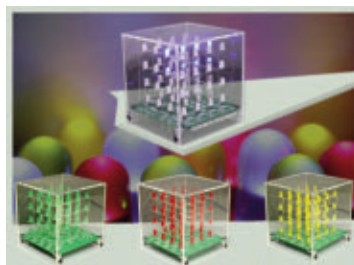
Pick a circuit!

With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.

Subscriber's Price \$32.95

Non-Subscriber's Price \$35.95

3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes. Colors available: Green, Red, Yellow & Blue

Subscriber's Price \$57.95

Non-Subscriber's Price \$59.95

Radiation Monitor Alarm Kit



This is an inexpensive surface-mount project that is good for beginners to start with. This kit has its own printed circuit board (PCB) which makes mounting the components easy. Plus, it comes in a pocket size shielded aluminum case measuring 3.5" in length and 1" in diameter. The on-off switch is a pushbutton on the bottom.

Subscriber's Price \$32.45

Non-Subscriber's Price \$33.95

Geiger Counter Kit

As seen in the March 2013 issue.



This kit is a great project for high school and university students. The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr. The LND712 tube in our kit is capable of measuring alpha, beta, and gamma particles.

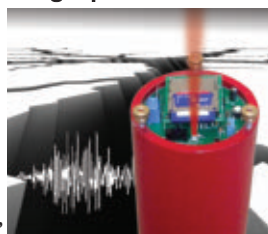
Partial kits also available.

Subscriber's Price \$145.95

Non-Subscriber's Price \$148.95

Seismograph Kit

As seen in the May 2012 issue.



Now you can record your own shaking, rattling, and rolling.

The Poor Man's Seismograph is a great project /device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

Subscriber's Price \$79.95

Non-Subscriber's Price \$84.95

Battery Marvel

As seen in the November 2011 issue.



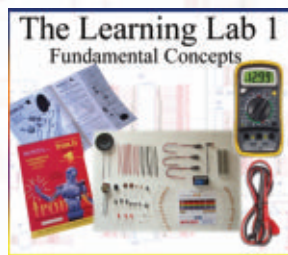
Battery Marvel helps protect cars, trucks, motorcycles, boats, and any other 12V vehicles from sudden battery failure. This easy-to-build kit features a single LED that glows green, yellow, or red, indicating battery health at a glance. An extra-loud piezo driver alerts you to any problems.

Details, please visit our website.

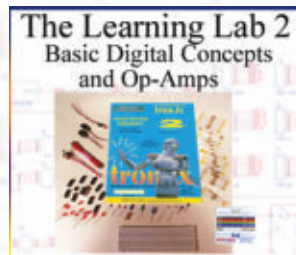
Subscriber's Price \$18.95

Non-Subscriber's Price \$19.95

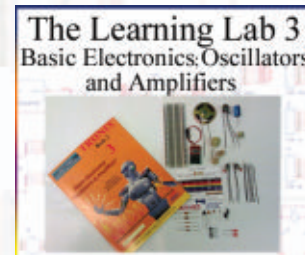
FOR BEGINNER GEEKS!



\$59.95



\$49.95



\$39.95

The labs in this series — from GSS Tech Ed — show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and a promotional video, please visit our webstore.

CLASSIFIEDS

SURPLUS

SURPLUS ELECTRONIC PARTS & ACCESSORIES

Over 20,000 Items in Stock.

Cables	Hardware	Relays	Switches
Connectors	LEDs	Semiconductors	Test Equipment
Displays	Motors	Service Manuals	Tools
Fans	Potentiometers	Speakers	VCR Parts

Surplus Material Components
SMC ELECTRONICS
www.smcelectronics.com

No Minimum Order.
Credit Cards and PAYPAL Accepted.
Flat \$4.95 per order USA Shipping.

COMPONENTS

RF PARTS™ CO.

Tubes, Transistors, Power Components

Email: rfp@rfparts.com • Web: www.rfparts.com

800-737-2787 Fax 888-744-1943

uM-FPU64

64-bit Floating Point Coprocessor



DIP-28, SOIC-28, TQFP-44

64-bit and 32-bit IEEE 754 compatible
SPI or I2C interface, 3.3V Supply, 5V Tolerant I/O
Extensive floating point support, FFT, matrix operations
GPS input, local device support, Flash functions, IDE

Robotics
Navigation
Sensor Modules
Embedded Systems

Additional products...

uM-FPU V3.1 32-bit FPU
uM-PWM1 Servo Coprocessor

www.micromegacorp.com

CONTROLLERS

INTEGRATED Ethernet PLCs for OEMs



From \$119

NANO-10

- ETHERNET / Modbus TCP/IP
- 8 DI/Os, 2 AI/Os
- RS485
- PWM / PID / Stepper Control
- Ladder + BASIC Programming

tel : 1 877 TRI-PLCS
web : www.triplc.com/nvhtml



TRIANGLE
RESEARCH
INTERNATIONAL

SERVICES



Sandpiper Electronics Services LLC

"design of custom electronics"

Prototype electronics development for research, industry & inventors. Experience in laser systems, flight experiments & laboratory research applications.

www.sandtronics.com

ROBOTICS

MaxBotix™ High Performance Ultrasonic Rangefinders

Check us out Today!

HRXL MaxSonar® WR™
High noise tolerance
IP67 rated
1 mm resolution
Multi Sensor operation
Calibrated beam pattern
Starting at \$109.95



XL MaxSonar® EZ™

Great for UAV's and robotics
Incredible noise immunity
Small in size
1cm resolution
Automatic calibration
Starting at \$39.95



Phone: 218 454 0706 Email: sales@maxbotix.com

www.maxbotix.com

MOBILE SCOOTERS



Sunny Solutions
Sales and Service

Need Help Getting Around?
Elec Mobile Scooters

www.ScootersByTom.com

800 315-0309

KITS/PLANS



QKITS LTD
sales@qkits.com

1 888 GO 4 KITS

Arduino • Raspberry Pi
Power Supplies
MG Chemicals
RFID



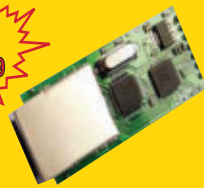
Visit us at:

www.qkits.com

ELECT DATA SYSTEMS

\$6000: TTL to Ethernet Converter

Only
\$17.99



- Auto detect 10/100M High speed Ethernet
- Baud rate 300 ~ 25600 bps
- Agreement: ETHERNET, ARP, IP, UDP, TCP
- Can work as Virtual COM
- 3.3V or 5V power input interface

Ethernet, Zigbee, RS485 IO modules at
WWW.SHJELECTRONIC.COM

MISC FOR SALE

HUGE CLOSE-OUT SALE!

All kinds of electronic products and parts!

Test equipment, schematics, tubes, audio-video equipment ... too numerous to list. Call for availability.

loustvelectronics@verizon.net

215.271.1138 or 215.468.3028

215.432.0333

Call Anytime/24-7

5-Digit LED Display

For the Basic Stamp and other micro-controllers.

Uses the MC14489 chip

\$28.99 w/FREE SHIPPING

See our web site for details.

www.boatanchoraudio.com

LIGHTING



Automatic
Stair Lighting
www.ReactiveLighting.com

WANT TO BUY



Excess Solutions
156 S. Milpitas Blvd.
Milpitas, CA 95035

Come in and see our retail store!

We Buy & Sell Excess Electronic Inventory

(408) 262-3900

www.excesssolutions.com

HARDWARE WANTED

DEC EQUIPMENT WANTED!!!

Digital Equipment Corp.
and compatibles.
Buy - Sell - Trade

CALL KEYWAYS 937-847-2300
or email buyer@keyways.com

AUDIO/VIDEO PARTS



OVER 15,000
ELECTRONIC PARTS
IN STOCK



Speakers



Components



Project Accessories



Tools and Tech Aids

Call or visit us online today to receive your FREE copy of our 2014 catalog!

parts-express.com/nuts

1-800-338-0531

ICEpower MODULES

50W x 2 (50ASX2) or
170W x 1 (50ASX2BTL)
Just add AC & speaker(s)
\$99 USA \$119 Worldwide
www.classicaudioparts.com

www.nutsvolts.com

>>> QUESTIONS

555 Anomaly

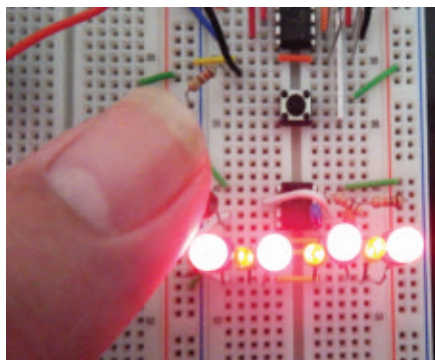
I have found an interesting anomaly with the 555 IC. Everywhere I've read and looked, the #4 (reset) pin has voltage held high to allow the IC to conduct, and then dropped low to turn off (reset) the IC.

I have 8.58V at 39.5 mA at VCC (#8) pin when the IC conducts. As you can see in the **schematic**, I am controlling the IC through the reset pin (#4) with a CdS photocell photoresistor. When light is removed from the CdS photocell and the IC conducts, it sends 6.5V at 18.2 mA to the output (#3) pin and LEDs. Apply light to the CdS photocell and the IC stops conducting; you then get -0.2V at 0.0 mA at the output (#3) pin. Since I added a 4.7 μ F electrolytic capacitor to the output pin, the IC now conducts 7V to the LEDs.

I am an electronics hobbyist who has been learning electronics for one year. I welcome any comments and ideas anyone has about what is going on with the IC.

#4141

Robert Calk Jr.
via email



Arduino Power Requirements

I'm using an external 6V supply to power an Arduino circuit which is giving me problems. I heard somewhere that the Arduino requires at least 7 or 8 VDC input. Doesn't the Arduino run on 5V? Can someone clarify?

#4142

Arturo Martinez
via email

How Much Heat Can A Battery Take?

I need to shrink wrap sets of AA lithium batteries for a radio project. Can I safely heat the shrink wrap with a hot air blower, without harming or causing the batteries to explode?

#4143

James Cook
Lansing, MI

Receiver ... Not

I inherited an old tube-type short wave receiver from my grandfather. It produces nothing but static, even after I replaced all of the vacuum tubes. All of the switches are good. Any ideas where to go from here?

#4144

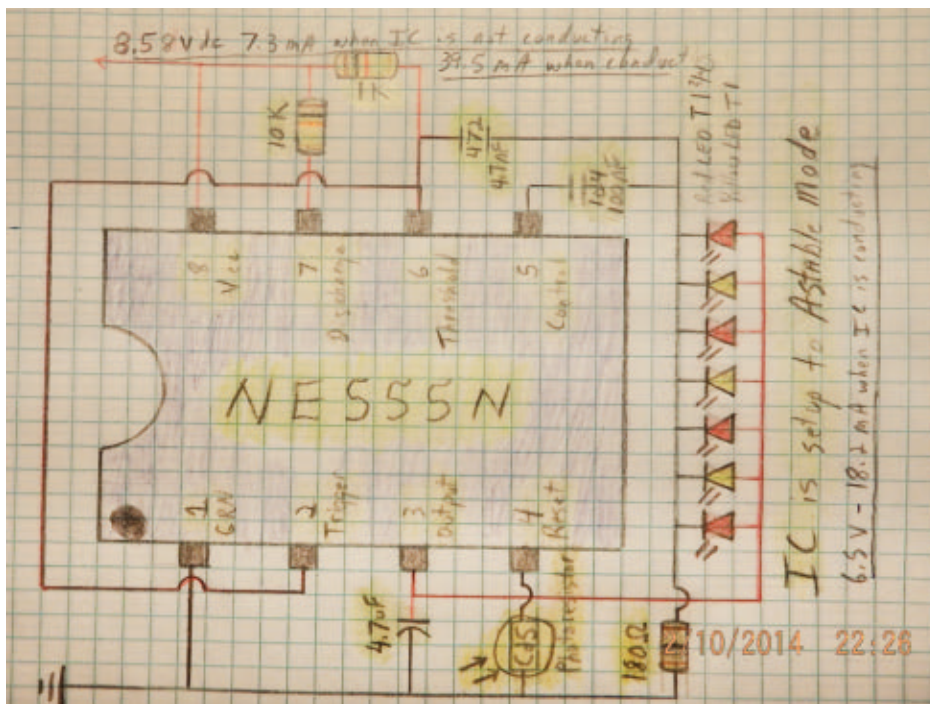
Ben Hill
Norfolk, VA

Self-Dimming Light Switch

Can you give me the schematic for a light switch that automatically dims the lights between, say, 10 pm and 6 am? I want something in between total darkness and blazing brightness for raiding the kitchen refrigerator in the middle of the night.

#4145

Alfonzo Garcia
Little Rock, AR



Send all questions and answers by email to forum@nutsvolts.com
or via the online form at www.nutsvolts.com/tech-forum

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All

submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES**

WHATSOEVER are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

Send all questions and answers by email to forum@nutsvolts.com
or via the online form at www.nutsvolts.com/tech-forum

Laser Power Supply

eBay had an amazing deal on high-power IR lasers. I want to build a laser cutting tool with one of the laser diodes. The heatsink isn't a problem, but I'm having trouble designing a constant current power supply that can handle the 50W laser. The commercial power supplies are hundreds of dollars. Any suggestions?

#4146

Carl Edwards
Santa Fe, NM

produce nearly anything you can imagine as long as it fits into the 3D printer. I have seen a complete ball bearing set made as one piece (no assembly required) on a old 3D printer which moved exactly like a metal bearing produced by sophisticated

machining processes, but the plastic did not have the strength to hold up in service. I can see using ceramic materials which can be fired to produce strength as a future wave in 3D printing.

Tim Brown PhD EE, PE
via email

>>> ANSWERS

[#2142 - February 2014]

Laser Cutter To 3D Printer

I want to move from a mill to a 3D printer to fabricate parts. As far as printing materials go, I've heard that regular plastic is toxic and print quality is poor, and the PLA alternative is brittle and heavy. What's the best printing material out there? Are there better choices?

3D printers (a.k.a., rapid prototyping machines) and laser cutters are very different animals. The laser cutter can be used to "machine" metals by laser ablation of the surfaces. A 3D printer (such as MakerBot) uses a curable plastic powder or filament for producing layer upon layer renditions of whatever you draw on a Computer Aided Drawing (CAD) system.

Polyethylene Terephthalate (PET) filaments produce stronger products than the usual ABS plastic materials. (A UT Austin student reportedly made a working pistol out of ABS but I am not sure I would want to trust my safety to the strength of plastic in this case.)

The replicator (3D printer) will run about \$3,000. MakerBot also sells a digitizer for about \$1,000 which allows you to reproduce an existing product by scanning it into the computer versus the laborious process of producing a 3D CAD drawing. Using the replicator and the digitizer/CAD system together makes it possible to



AUVSI's
Unmanned Systems
2014

REGISTER TODAY



CONFERENCE 12-15 MAY
TRADE SHOW 13-15 MAY

ORANGE COUNTY CONVENTION CENTER | ORLANDO, FLA. | USA

REGISTER BY 15 MARCH AT AUVSISHOW.ORG AND SAVE

 **AUVSI**
ASSOCIATION FOR UNMANNED
VEHICLE SYSTEMS INTERNATIONAL

AUVSISHOW.ORG

NEW PRODUCTS

Continued from page 19



power supplies), and products with variable-speed motor control (e.g., fans, home appliances).

The PIC16(L)F161X family is supported by Microchip's standard suite of development tools, including the MPLAB® ICD 3 (part # DV164035, \$189.99) and PICkit™ 3 (part # PG164130, \$44.95) programmer/debuggers, along with the PICkit™ 3 Low Pin Count Demo Board (part # DM164130-9, \$25.99).

For more information, contact:
Microchip Technology, Inc.
Web: www.microchip.com

LOW PIN COUNT, GENERAL-PURPOSE EIGHT-BIT PIC FAMILY **EXPANDED**

Microchip Technology Inc., has announced expansion of its eight-bit PIC microcontroller (MCU) portfolio, with the peripheral-rich, low pin count PIC16(L)F161X family. These new MCUs expand the offering of Microchip's Core Independent Peripherals (CIP) which offload timing-critical and core-intensive tasks from the CPU, allowing it to focus on other application tasks. Additionally, this family integrates fault-detecting hardware features to assist engineers in developing safety-critical applications.

The PIC16(L)F161X family offers a variety of key features, including the Windowed Watchdog Timer (WWDT) which monitors proper software operation within predefined limits, improving reliability. The Cyclic Redundancy Check with Memory Scan (CRC/SCAN) detects and scans memory for corrupted data.

This family also includes a Hardware Limit Timer (HLT) which detects hardware fault conditions (stall, stop, etc.), simplifying closed-loop control applications. These peripherals make it easier for designers to implement safety standards (e.g., UL and class B) or fail-safe operation.

In addition to the HLT, the PIC16(L)F161X features the unique, 24-bit Signal Measurement Timer (SMT). The SMT performs high-resolution measurements of any digital signal in hardware, resulting in more precise and accurate measurements. This is ideal for speed control, range finding, and RPM indicators. Both timers are designed to reduce design complexity by eliminating the need for additional code and external components.

This high level of integration makes these MCUs appealing to a broad range of applications, such as monitoring and fail-safe systems (e.g., industrial machinery,

THREE-WHEEL ROBOT PLATFORM KIT

ServoCity is now offering a new three-wheel robot platform. This kit includes all the mechanical parts needed to build an omni-directional robot. Complete with



large 6" diameter drive wheels with rubber lips, it offers fantastic traction. The ball bearing swivel along with the ball bearing supported 4" wheel at the rear enables smooth turning transitions.

The chassis can easily be reconfigured for various applications and accepts all Actobotics™ components. The included motor mounts are compatible with ServoCity's 3V-12V gearmotors and 6V-12V precision gearmotors (drive motors sold separately). Users can incorporate many types of drive motors by simply switching out the motor mounts. Retail price is US\$79.99.

For more information, contact:
ServoCity
Web: www.servocity.com

AMATEUR RADIO

AND TV

Ramsey Electronics82-83

BATTERIES/CHARGERS

Cunard Associates19

DIGITAL

OSCILLOSCOPES

Rigol Technologies6

BUYING ELECTRONIC

SURPLUS

Weirdstuff Warehouse19

CCD CAMERAS/VIDEO

Ramsey Electronics 82-83
Redwood Electronics Corp19

CIRCUIT BOARDS

AP Circuits15
Cunard Associates19
Digilent5
Dimension Engineering.....33
ExpressPCB55
Front Panel Express LLC50
GHI Electronics33
PCB Pool33
Saelig Co. Inc.Back Cover

COMPONENTS

Eastern Voltage Research26
Redwood Electronics Corp19
SDP/SI19

COMPUTER

Hardware

Weirdstuff Warehouse19

Microcontrollers /

I/O Boards

GHI Electronics33
M.E. Labs.....44

MikroElektronika3

Parallax, Inc.14

Technologic Systems37

DESIGN/ENGINEERING/

REPAIR SERVICES

ExpressPCB55
Front Panel Express LLC50
PCB Pool33

DRIVE COMPONENT

CATALOGS

SDP/SI19

EDUCATION

Command Productions7
Digilent5
NKC Electronics19
Poscope66
UCount14

EMBEDDED TOOLS

NetBurner2

ENCLOSURES

Front Panel Express LLC50

EVENTS

AUVSI79
Maker Faire.....51

KITS & PLANS

Eastern Voltage Research26
NetBurner2
NKC Electronics19
QKITS19
Ramsey Electronics82-83
UCount14

MISC./SURPLUS

All Electronics Corp.13
Front Panel Express LLC50
Weirdstuff Warehouse19

MOBILE RADIO

MOUNTS

E Tip Inc.19

PROGRAMMERS

M.E. Labs.....44
MikroElektronika3

RF TRANSMITTERS/

RECEIVERS

E Tip Inc.19

ROBOTICS

Digilent5
GHI Electronics33
Lemos International Co.15
SDP/SI19
UCount14

SATELLITE

Lemos International Co.15

SECURITY

E Tip Inc.19

TEST EQUIPMENT

Dimension Engineering.....33
NKC Electronics19
Pico Technology27
Poscope66
Rigol Technologies6
Saelig Co. Inc.Back Cover

TOOLS

MikroElektronika3
NetBurner2
PanaVise50
Poscope66

WAVEFORM

GENERATORS

Rigol Technologies6

All Electronics Corp.13

AP Circuits15

AUVSI79

Command Productions7

Cunard Associates19

Digilent5

Dimension Engineering.....33

Eastern Voltage Research ..26

E Tip Inc.19

ExpressPCB55

Front Panel Express LLC ...50

GHI Electronics33

Lemos International Co.15

Maker Faire.....51

M.E. Labs.....44

MikroElektronika3

NetBurner2

NKC Electronics19

PanaVise50

Parallax, Inc.14

PCB Pool33

Pico Technology27

Poscope66

QKITS19

Ramsey Electronics82-83

Redwood Electronics Corp .19

Rigol Technologies6

Saelig Co. Inc.Back Cover

SDP/SI19

Technologic Systems37

UCount14

Weirdstuff Warehouse19



Super-Pro FM Stereo Radio Station

SPRING SALE!

- ✓ PLL synthesized for drift-free operation
- ✓ Built-in mixer - 2 line inputs and one microphone input, line level monitor output!
- ✓ Frequency range 88.0 to 108.0, 100 kHz steps
- ✓ Precision active low-pass "brick wall" audio filter!
- ✓ Dual LED bar graph audio level meters!
- ✓ Automatic adjustable microphone ducking!
- ✓ Easy to build through-hole design!



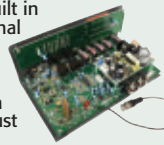
This professional synthesized transmitter is adjustable directly from the front panel with a large LED digital readout of the operating frequency. Just enter the setup mode and set your frequency. Once selected and locked you are assured of a rock stable carrier with zero drift. The power output is continuously adjustable throughout the power range of the model selected. In addition, a new layer of anti-static protection for the final RF amplifier stage and audio inputs has been added to protect you from sudden static and power surges.

Audio quality is equally impressive. A precision active low-pass brick wall audio filter and peak level limiters give your signal maximum "punch" while preventing overmodulation. Two sets of rear panel stereo line level inputs are provided with front panel level control for both. Standard unbalanced "RCA" line inputs are used to make it simple to connect to the audio output of your computer, MP3 player, DVD player, cassette deck or any other consumer audio source. Get even more creative and use our K8094 below for digital storage and playback of short announcements and ID's! In addition to the line level inputs, there is a separate front panel microphone input. All three inputs have independent level controls eliminating the need for a separate audio mixer! Just pot-up the source control when ready, and cross fade to the 2nd line input or mic! It's that simple! In addition to the dual stereo line inputs, a stereo monitor output is provided. This is perfect to drive studio monitors or local in-house PA systems.



The FM100B series includes an attractive metal case, whip antenna and built in 110/220VAC power supply. A BNC connector is also provided for an external antenna. Check out our Tru-Match FM antenna kit, for the perfect mate to the FM100B transmitter.

We also offer a high power kit as well as an export-only assembled version that provides a variable RF output power up to 1 watt. The 1 watt unit must utilize an external antenna properly matched to the operating frequency to maintain a proper VSWR to protect the transmitter.



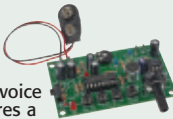
(Note: The FM100B and FM100BEX are do-it-yourself learning kits that you assemble. The end user is responsible for complying with all FCC rules & regulations within the US or any regulations of their respective governing body. The FM100BWT is for export use and can only be shipped to locations outside the continental US, valid APO/FPO addresses or valid customs brokers for documented end delivery outside the continental US).

FM100B Super-Pro FM Stereo Radio Station Kit, 5uW to 25mW Output
FM100BEX Super-Pro FM Stereo Radio Station Kit, 5uW to 1W Output

\$239.95
\$299.95

Digital Voice Changer

This voice changer kit is a riot! Just like the expensive units you hear the DJ's use, it changes your voice with a multitude of effects! Features a built-in microphone and both a speaker and line output. Runs on a standard 9V battery, not included.



MK171 Digital Voice Changer Kit \$14.95

Audio Recorder & Player

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.



K8094 Audio Recorder/Player Kit \$32.95

Passive Aircraft Monitor

PATENTED!

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.



ABM1 Passive Aircraft Receiver Kit \$89.95

Laser Trip Sensor Alarm

True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.



LTS1 Laser Trip Sensor Alarm Kit \$29.95

Collinear Vertical FM Antenna

Our 5/8 wave omni antenna has been the standard for LPFM installations worldwide. Provides 3.4dB gain while keeping the signal radiation low to the horizon for maximum range. Field tuneable over the entire FM range for a perfect match. SO239 connector for PL259 plug.



FMA200E Omnidirectional FM Antenna \$119.95

Logic Interface Module

Interface your digital output to the real world with an on-board SPDT relay rated at 240V at 10A! It takes a digital low (.5VDC or less) or a high (+1 to +12VDC) and provides your choice of an active low or high closure! It's that simple! Runs on 12VDC at 60mA.



RI1 Logic Interface Kit \$17.95

4-Channel USB Relay Control

This professional quality USB relay controller allows computer controlled switching of external devices, plus full bi-directional communication with the external world using the USB port of your computer!



The controller features four onboard relay outputs with a current rating of 10A each. Also onboard is a 6-channel Input/Output interface, with each channel individually configurable as Digital Input, Digital Output, Analog Input (10-bit Resolution). In Digital Input/Output modes, each channel can support a TTL compatible or ST input or a 5V output signal. In Analog Input mode, each channel can convert a voltage of between 0 to 5V into a 10-bit digital representation.

UK1104 4-Ch USB Relay Interface Kit \$59.95

Classic Nixie Tube Clocks



Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more!

The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment.

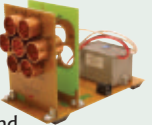
Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of!

The clocks are available in our signature hand rubbed Teak & Maple, polished stainless, or clear acrylic bases. You also have your choice of IN-14 or highly sought after IN-8-2 nixie tubes (for the 6-tube clock).

NIXIE Classic Nixie Tube Clock Kits from \$229.95

Air Blasting Ion Generator

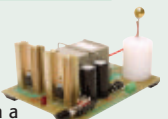
Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.



IG7 Ion Generator Kit \$64.95

HV Plasma Generator

Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.



PG13 HV Plasma Generator Kit \$64.95

Signal Magnet Antenna

The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.



SM100 Signal Magnet Antenna Kit \$89.95

Broadband RF Preamp

Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.



PR2 Broadband RF Preamp \$69.95

Active Receive Antenna

The popular antenna for the serious DX'ers works on all bands - shortwave, HF, VHF, and UHF yet performs like a 60' long wire antenna! Provides over 15dB of gain, and includes auto-off RF bypass and front panel gain control.



AA7C Active Antenna Kit \$59.95



There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit www.ramseycatalog.com

Follow Us and SAVE \$\$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!



Stereo Audio Gain Controller

- ✓ Stereo audio processing while preserving audio dynamics!
- ✓ True stereo control keeps virtual sonic source location intact!
- ✓ Auto-bypass restores original levels when power is turned off!
- ✓ Built-in bar graph indication of signal level with display mute!

The SGC1 is one of our latest kits, and provides a great solution to the age-old problem: how can we easily correct inconsistent audio levels without negatively affecting the dynamics of the audio signal? The SGC1 circuit implements a principle known as the "Platform Gain Principle," which was originally developed by CBS Labs (what we now know as CBS in the TV and radio world) to allow transmitted audio levels to be automatically adjusted to keep them within a desired range.

Think of it like an audio engineer, constantly adjusting the output level in order to limit highs that would be too loud while boosting lower levels so that they can still be heard. You may think "oh, this is just another limiter/compressor!" Not so! Here's the real trick: keeping the full dynamic range ratio of the output signal the same as the original input - something the typical limiter/compressor can only dream of doing! The SGC1 can be placed in just about any standard analog stereo line level audio circuit (the red and white RCA connectors or the mini-phone connector) to keep the audio level within the desired range. It's also the perfect addition to any of our hobby kit transmitters, allowing you to match levels between different audio sources while keeping lows audible and preventing the highs from overdriving. The SGC1 makes a great addition to any audio system where you need to keep levels from different sources under control, but still make sure they all sound great! In addition to its useful basic function and great audio performance, the SGC1 also boasts a front panel LED meter to give an indication of the relative level of the input signal, plus a level control (also on the front panel) that allows you to adjust the controller to the min/max center point of your desired level range. And yes, it is a **Stereo Gain Controller!** Meaning that the levels of both the left and right channels are monitored and adjusted equally, thereby maintaining the relative virtual position of things like instruments, singers and speakers! The entire unit is housed in a slim attractive black textured aluminum case that is sure to complement your studio or home theatre. If you're looking for perfect audio levels, hire a broadcast audio engineer, but if that doesn't fit your budget, the SGC1 is the next best thing! Includes 15VDC world-wide power adapter.

SGC1 Stereo Audio Platform Gain Controller Kit

\$179.95

8-Channel Remote Ethernet Controller

Now you can easily control and monitor up to 8 separate circuits via the standard Ethernet network in your home or office. Connection wise it couldn't be simpler. The controller functions as an IP based web server, so it can be controlled by any internet browser that can reach your network! There are no drivers or proprietary software required, just access the controller like any web page from your PC, laptop, or even your smartphone!

Security is assured allowing up to 4 separate user credentials. The controller can be set to a specific static IP within your network subnet or can be set to DHCP (auto negotiate). The controller can even be programmed to send you an email to notify and confirm power up and status changes!

To simplify the connection of your equipment to the controller, 8 separate and isolated relay outputs are provided! This gives you internet or network control of up to 8 separate functions. No need to worry about interfacing a logic high or logic low, or burning up the interface! The applications are endless! From something as simple as turning on and monitoring lights at your house with a normal latched closure to advanced control of your electronic gadgets, radio equipment, or even your garage door!

Each relay contact is rated at 12A at 30VDC or 16A at 230VAC. Each of the 8 channels has built-in timer and scheduler programs for day, weekend, working days, every day, and every day except Sunday. Relay control functions are programmable for on, off, or pulse (1-99 seconds, 1-99 minutes, or 1-99 hours). In addition to control functions, the web interface also displays and confirms the status of each channel. Each channel can be custom labeled to your specific function name. The controller operates on 12VDC or 12VAC at 500mA or our new AC121 global 12VDC switching power supply below. Factory assembled, tested, and ready to go! Even includes a Cat-5 cable!

VM201 8-Channel Remote Ethernet Controller, Factory Assembled & Tested

\$169.95

Tickle-Stick Shocker

The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

TS4 Tickle Stick Kit

\$9.95

Tri-Field Meter Kit

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of ghosts! Req's 4 AAA batteries.

TFM3C Tri-Field Meter Kit

\$74.95

Electret Condenser Mic

This extremely sensitive 3/8" mic has a built-in FET preamplifier! It's a great replacement mic, or a perfect answer to add a mic to your project. Powered by 3-15VDC, and we even include coupling cap and a current limiting resistor! Extremely popular!

MC1 Mini Electret Condenser Mic Kit

\$3.95

12VDC Regulated Switching Supply

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores for RF and EMI suppression. All this at a 10 buck odd wallwart price! What a deal!

AC121 12VDC 1A Regulated Supply

\$9.95

12VDC Worldwide Supply

It gets even better than our AC121 above! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility! Dual ferrite cores!

P529 12VDC 1.25A Global Power Supply

\$19.95

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 Sniff-It RF Detector Probe Kit

\$27.95

4-Channel USB Relay Control

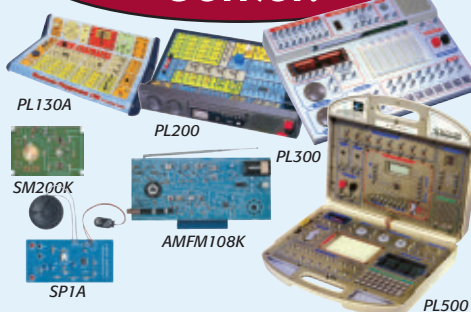
This professional quality USB relay controller and data acquisition module allows computer controlled switching of external devices, plus full bi-directional communication with the external world using the USB port of your computer. It is compatible with both Windows and Apple OS X, as well as various Linux flavors. When you plug it into your computer, it appears as a USB CDC device that creates a Virtual Serial (COM) port allowing easy communication with the board through any programming language that supports serial communications (VB, VB.NET, C#, C, C++, Perl, Java, etc).

The controller features four onboard relay outputs with a current rating of 10A each. Also onboard is a 6-channel Input/Output interface, with each channel individually configurable as Digital Input, Digital Output, Analog Input (10-bit Resolution), or DS18B20 series Temperature Sensor. In Digital Input/Output modes, each channel can support a TTL compatible or ST input or a 5V output signal. In Analog Input mode, each channel can convert a voltage of between 0 to 5V into a 10-bit digital representation. Finally, in Temperature Sensor mode, each channel can be connected to a DS18B20 series Digital Temp Sensor.

UK1104 4-Ch USB Relay Interface Kit

\$59.95

The Learning Center!



Fun Electronic Learning Labs

- ✓ Learn and build!
- ✓ 130, 200, 300, & 500 in one labs!
- ✓ Practical through hole and SMT soldering labs!
- ✓ Integrated circuit AM/FM radio lab!
- ✓ Super comprehensive training manuals!

Starting out our "All in One" series, the PL130A, gives you 130 different electronic projects, together with a comprehensive 162 page learning manual. A great start for the kids...young and old! Next, check out the PL200, that gives you 200 very creative and fun projects, and includes a neat interactive front panel with 2 controls, speaker, LED display and a meter. From there, step up to our PL300, which gives you 300 separate electronic projects along with a 165 page learning and theory manual. The PL300 walks you through the learning phase of digital electronics. If you're looking for the ultimate lab kit, check out our PL500. It includes a whopping 500 separate projects, a 152 page starter course manual, a 78 page advanced course manual, and a 140 page programming course manual! The PL500 covers everything from the basics to digital programming!

If you are looking to either learn or hone up on your through hole or SMT soldering skills check our SPIA and SM200K Practical Soldering Labs. You will be a soldering master in no time!

We make it easy to learn IC's while at the same time, building a neat AM/FM radio with our AMFM108K AM/FM IC Lab Kit. You will have a blast AND learn!

PL130A	130-In-One Lab Kit	\$39.95
PL200	200-In-One Lab Kit	\$84.95
PL300	300-In-One Lab Kit	\$109.95
PL500	500-In-One Lab Kit	\$249.95
SPIA	Through Hole Soldering Lab	\$9.95
SM200K	SMT Practical Soldering Lab	\$22.95
AMFM108K	AM/FM IC Lab Kit & Course	\$36.95

GET THE INUTS&VOLTS DISCOUNT!

Mention or enter the coupon code **NVRMZ142** and receive 10% off your order!

800-446-2295
www.ramseykits.com

Prices, availability, and specifications are subject to change. We are not responsible for typos, stupid, printer's bleed, or confusion that April showers bring May flowers! Robin thinks winter is over, just because she lives in CA! Wrong! Visit www.ramseykits.com for the latest pricing, specials, terms and conditions. Copyright 2014 Ramsey Electronics®... so there!

RAMSEY ELECTRONICS®

590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

100MHz 1GSa/s 10Mpt 2-Ch Oscilloscope



\$429

Deep Memory Scope!

Owon's SDS7102 is a 100MHz bandwidth scope with huge 10Mpt memory for fast signal investigations. Battery option for use away from 110V - great for field work!

"I never expected a high end scope for around \$400 but Owon delivers more features than I anticipated." -N.

"For the price this scope cannot be beat!" -D.

Your choice!

2-Channel Digital Storage Oscilloscopes with High-End Features:

High-performance digital scopes for education, training, production line, and R&D. 8" color 800x600 TFT-LCD screen, AutoScale function to simplify use. DB-15 VGA port shows screen display on external monitors/projectors - ideal for teaching. USB flash disk storage, Pass/Fail, LAN for remote measurements. Sophisticated triggering includes: Edge, Pulse, Video, and Slope. Waveform recording and replay.

30MHz 250MSa/s 2-Ch Oscilloscope



\$289

Lowest Cost Scope!

Owon's SDS5032E is an economical 2-ch 30MHz 250MSa/s digital storage oscilloscope with 10Kpt memory. Excellent starter scope!

"Great scope for the price. Love this scope, especially the clear full color 800x600 screen." -A.

"...nice variety of features and unmatched for price." -Z.

www.saelig.com